НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки
Кафедра Кафедра автоматики і управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ Ролік О.І.
(підпис) (ініціали, прізвище)
«___»_____20__р.

Дипломний проект

на здобуття ступеня бакалавра

зі спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

на тему: Система моніторингу стану домашніх тварин

Виконав: студент 4 курсу, групи ІА-61
(шифр групи)
Капущак Анна Степанівна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник роботи: ст. викладач Шимкович Володимир Миколайович _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання) (підпис)

Рецензент: к.т.н., доцент, доцент кафедри ОТ Волокита А.М _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент_____
(підпис)

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет Інформатики та обчислювальної техніки

Кафедра Автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Напрям 151 Автоматизація та комп'ютерно-інтегровані технології

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Ролік О.І.

(підпис) (ініціали, прізвище)

« __»_____2020р.

ЗАВДАННЯ

на дипломной проєкт студенту

Капущак Анні Степанівні

(прізвище, ім'я, по батькові)

1. Тема проєкту: Система моніторингу стану домашніх тварин

Керівник проєкту: ст. викладач кафедри АУТС Шимкович Володимир Миколайович

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «06» березня 2020р. №

_____

2. Строк подання студентом роботи

_____

3. Вихідні дані до роботи система моніторингу стану домашніх тварин, яка дозволяє збирати дані про стан здоров'я тварини та обробляє їх у зручному для користувачів форматі веб-додатку

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) вступ, призначення та область застосування, огляд та аналіз існуючих рішень, розробка схеми електричної структурної, розробка схеми електричної принципової, моделювання та конструювання додатку, висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) структурна схема, принципова схема, UML діаграма класів, структура бази даних

---

6. Дата видачі завдання 07.03.2020

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд та аналіз існуючих рішень	07.03 - 14.03	
2	Розроблення структурної схеми	15.03 - 28.03	
3	Розроблення принципіальної схеми	29.03 - 12.04	
4	Вибір окремих елементів	13.04 - 19.04	
5	Проєктування додатку	20.04 - 03.05	
6	Створення бази даних та серверу	04.05 - 17.05	
7	Розробка інтерфейсу системи	18.05 - 24.05	
8	Створення інфструкції користувача	25.05 - 31.05	
9	Оформлення документації проєкту	01.06 - 11.06	

Студент _____ Капущак А.С..
(підпис) (ініціали, прізвище)

Керівник роботи _____ Шимкович В.М.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Капущак А.С. «Система моніторингу стану домашніх тварин». НТУУ «КПІ ім. Ігоря Сікорського», Київ, 2020.

Дипломний проект присвячений розробці автономної системи моніторингу стану домашніх тварин. Система з неінвазивних датчиків збирає дані про стан здоров'я, а саме, пульс та температуру. Також система відслідковує локацію, шлях пересування та безпечні зони. Дані зберігаються та обробляються в розробленому веб-додатку, в якому є змога додатково виводити графіки динаміки показників здоров'я, отримувати сповіщення про критичні для здоров'я тварини показники та відстежувати маршрути пересування, поточне місцезнаходження.

Текстова документація містить інформацію про призначення, проектування,огляд існуючих рішень, розробку, та особливості роботи із системою. Велика увага була приділена вибору технічних засобів для реалізації та проектуванню додатку системи.

Ключові слова: система моніторингу, датчик, додаток, програма.

Розмір пояснювальної записки – 68 аркушів, містить 29 ілюстрації, 13 таблиць.

ANNOTATION

Kapushchak Anna «Pet health and activity monitoring system».. KPI, Igor Sikorsky's name, Kyiv, 2020.

The diploma project is devoted to the development of an autonomous system for monitoring pets health and activity

The system uses non-invasive sensors to collect health data, such as heart rate and temperature. The system also tracks location, movement and safe areas. The data is stored and processed in a developed web application allows users to display graphs of the dynamics of health indicators, receive notifications about critical indicators for the health of the animal and track the routes of movement and current location.

Text documentation contains information about the purpose, projection, review of existing solutions, development, and features of the system. Much attention was paid to the choice of technical means for the implementation and projection of the system application.

Key words: monitoring system, sensor, application, program.

The size of the explanatory note is 68 sheets, contains 29 illustrations, 13 tables

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2			Знову розроблена			
3						
4						
5	А4	ІА61.100БАК.005 ПЗ	Пояснювальна записка	69		
6	А3	ІА61.100БАК.005 Э.1	Структурна схема	1		
7	А3	ІА61.100БАК.005 Э3	Схема електрична принципова	1		
8	А3	ІА61.100БАК.005.Д1	UML Діаграма класів	1		
9	А3	ІА61.100БАК.005.Д2	Діаграма бази даних	1		
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	**ІА61.100БАК.005 ТП**		
Розроб.		Капущак А.С.			**Відомість технічного проекту** **Система моніторингу стану домашніх тварин**	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Шимкович В.М.				Т	
Реценз.						НТУУ «КПІ ім. І. Сікорського» ФІОТ група ІА-61	
Н. Контр.							
Затв.							

Пояснювальна записка

до дипломного проєкту

на тему: «Система моніторингу стану домашніх тварин»

Київ – 2020 рок

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 4
ВСТУП 5
ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ 8
1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ 9
1.1 Огляд та аналіз існуючих систем 9
1.1.1 Система моніторигу PetPace Smart Collar 9
1.1.2 Система моніторингу FitBark 10
1.1.3 Порівняння обраних систем моніторингу 11
1.2 Огляд та аналіз існуючих додатків 12
1.2.1 Додаток PetPace Smart Collar 12
1.2.2 Додаток FitBark 14
1.2.3 Порівняння обраних додатків 15
Висновки до розділу 16
2 РОЗРОБКА СХЕМИ ЕЛЕКТРИЧНОЇ СТРУКТУРНОЇ 17
Висновки до розділу 23
3 РОЗРОБКА СХЕМИ ЕЛЕКТРИЧНОЇ ПРИНЦИПОВОЇ 24
3.1 Вибір окремих елементів 24
3.1.1 Акселерометр та гіроскоп 25
3.1.2 Мікроконтролер 26
3.1.3 Цифровий датчик температури 27
3.1.4 GSM модуль 28
Висновки до розділу 29
4 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ДОДАТКУ 30
4.1 Мова програмування та платформа 32
4.1.1 Бібліотеки використані для back-end розробки 33
4.2 Проєктування додатку 33

4.3 Опис створеної бази даних 35
4.4 Опис серверної частини системи 37
4.5 Розробка модуля відстежування 39
4.6 Опис розробленого інтерфейсу системи 41
4.7 Компоненти 44
4.8 Опис класів та методів системи 46
4.9 Інструкція користувача 52
Висновки до розділу 59
ВИСНОВКИ 60

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GPS - Global Positioning System.

API - Applicaфtion Programming Interface.

МК - мікроконтролер.

GSM - Global System for Mobile Communications.

ЕЗ - Ефект Зеєбека.

JS - JavaScript.

MVVM - Model-View-ViewModel.

БД - база даних.

СУБД - система управління базами даних.

SQL - structured query language.

ВСТУП

Можливість визначення розташування є невід'ємною частиною нашого життя. Типових застосувань цієї функції може бути досить багато: пошук загублених людей, послуги навігації, контроль переміщення автотранспорту і персоналу, послуги інформування про погодні та транспортні умови.

Найбільш розповсюджене застосування даної функції в поєднанні з іншими опціями, реалізовано в популярних гаджетах: розумних годинниках, фітнес-трекерах (трекери активності) та смартфонах. За допомогою цих пристроїв користувачі можуть відслідковувати свою фізичну активність та стан здоров'я, вимірювати пульс, температуру та інші параметри.

Звичайний трекер активності - це пристрій моніторингу та відстеження показників, таких як пройдена відстань, пульс та спожиті калорії. Деякі пристрої також можуть фіксувати географічне положення користувача (GPS), температуру тіла і середовища.

Слід зауважити, що більшість трекерів не мають свого програмного забезпечення для аналізу даних, а можуть лише фіксувати дані в форматі реального часу. Спеціальний додаток значно збагачує функціонал пристрою, адже допомагає не лише зберігати, але й аналізувати та систематизовувати отримані дані.

Розглянути дану систему можна не лише по відношенню до людей, але і тварин. Це досить корисна система, як для господарів собак чи котів, так і для власників ферм. Мова йде про будь-яку домашню тварину: вівці, корови, коні, собаки, коти та інші тварини. Сучасні технології вже повноцінно стали частиною фермерства і відіграють все більшу роль в його розвитку. В останні роки результати досліджень і розробок стають більш точними, одним з найбільших напрямів розвитку стало електронне тваринництво. Багато досліджень зосереджені саме на розвитку здоров'я тварин.

Подібний трекер дозволить знайти тварину, якщо вона загубилась, зникла з поля зору і тд. Така система дозволить власнику в режимі реального часу спостерігати за станом здоров’я та місцезнаходженням домашньої тварини.

Основними параметрами для моніторинга є пульс, температура та локація. Корисним доповненням служить додаток, який може аналізувати та будувати діаграми згідно отриманим даним. Для господарів котів чи собак, така динаміка може бути корисною для простого моніторингу, за станом свого улюбленця, поки вони на роботі. У випадку ж фермерів, це дозволить виділити найбільш здорову худобу та відслідковувати головні показники стану стада.

Моніторинг здоров'я залежить від двох методів, таких як прямий контакт (інвазивний) або непрямий контакт (неінвазивний). Тож реалізувати систему можна за допомогою неінвазивних датчиків в поєднанні з мікропроцесором для обробки інформації.

Метою дипломного проєкту є розробка системи моніторингу стану домашних тварин та створення програмного забезпечення для неї.

З ціллю досягнення поставленої мети, потрібно розглянути та провести аналіз існуючих рішень. Базуючись на результатах аналізу знайти шляхи модернізації та покращення існуючих систем. Також необхідно розробити технічні засоби, що дозволять вирішити поставлене завдання:

- розроблення електричної структурної системи;
- розроблення електричної функціональної системи;
- вибір окремих елементів для системи;
- вибір платформи та мови програмування;
- створення бази даних;
- створення серверної частини додатку;

- моделювання та проектування web-додатку.

Розглянуті рішення повинні забезпечувати моніторинг стану домашніх тварин, а саме: показників пульсу, температури та місцярозташування. Розроблений додаток має надавати користувачц можливість підключати системи моніторингу до додатку в необмеженій кількості – створювати тварин, зберігати та обробляти отриману інформацію з датчиків, створювати графіки в залежності від обраних користувачем параметрів, додавати безпечні локації на мапу програми.Додаток має бутиреалізований в зручному та простому для користувача форматі.

ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Створення вдосконалених систем трекінгу проводиться для зменшення використання людських ресурсів в деяких процесах, економії часу та забезпечення віддаленого слідкування за тими, чи іншими об’єктами.

В сучасному ритмі життя, важко поєднувати різноманітні сфери життя якісно, не втрачаючи важливих аспектів. Задачею даної системи моніторингу за станом домашніх тварин - є полегшити цю місію. Користувач може в режимі реального часу слідкувати за розміщенням свого домашнього улюбленця та бути впевненим, що йому нічого не загрожує і всі життєво важливі показники в нормі. Коли тварина хворіє також зручно стежити за її станом, поки господар на роботі. Зручний додаток дозволить вивести графік зміни стану тварини за день, тиждень чи навіть місяць.

Розробка може бути корисною також для фермерів, адже випасання худоби відбувається на величезних ділянках землі. Тварина може загубитись чи зайти надто далеко. Додаток дозволить з легкістю відслідкувати все стадо на карті, що значно полегшить пошук худоби.

1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд та аналіз існуючих систем

На сьогоднішній день у широкому доступі існує безліч різноманітних систем моніторингу за станом домашніх тварин. Кожна з них унікальна за своїм функціоналом, ціновою політикою та доступністю в окремих регіонах. Системи відрізняється своєю локалізацією та розрахована на окремі потреби користувачів. Деякі з них більш прості і розраховані лише для відслідковування розташування чи моніторингу стану здоров'я. Більшість систем не мають свого мобільного чи комп'ютерного додатку для збору та аналізу даних. Відображення даних відбувається на спеціальних пристроях-навігаторах, які показують карту місцевості в якій знаходиться тварина. Також пристрої відрізняються автономністю роботи, часом роботи за ряду, відстанню працездатності та алгоритмами передачі інформації. Найдорожчі системи мають власні багатофункціональні додатки з широким спектром можливостей. Тож до розгляду були обрані дві системи з різних сегментів доступності, які все ж є найпопулярнішими в своїй секції: PetPace Smart Collar і FitBark. Одним з найважливіших критеріїв до вибору системи стала наявність власного, сучасного додатку системи.

1.1.1 Система моніторигу PetPace Smart Collar

Смарт-ошийник PetPace - це бездротовий ошийник, який постійно збирає інформацію про життєві ознаки та поведінку домашньої тварини. Система PetPace Smart Collar зображена на рисунку 1.1. PetPace використовує неінвазивні датчики для відстеження температури, активності, пульсу, дихання, позицій, споживаних і спалених калорій, а також зміни частоти серцевих скорочень - всі показники, що підкреслюють загальний стан здоров'я тварини.

Ошийник надсилає дані через 2-, 15- або 30-хвилинні інтервали. PetPace працює зі спеціальним шлюзом, який підключається до вашого модему через порт Ethernet, та в будь-який час, коли тварина знаходиться в межах 1000 футів від шлюзу, ошийник передає дані.

Рисунок 1.1- Система PetPace Smart Collar

1.1.2 Система моніторингу FitBark

FitBark - один з найдрібніших у світі бездротовий трекерів активності для собак. Система FitBark зображена на рисунку 1.2.

Ця платформа, використовує існуючі API фітнес-трекерів людини і створює єдине уявлення про рівень активності домашнього улюбленця та господаря.

Система дає можливість отримувати оновлення про стан тварини в реальному часі, навіть поки вона не перебуває в межах Bluetooth на вашому девайсі. Користувач може відслідковувати зміни даних через Wi-Fi, адже дані в режимі онлайн завантажуються на веб-сервер.

Трекер може відслідковувати рівень активності, якість сну, пройдену відстань, спалювані калорії та загальне самопочуття 24-7.

Рисунок 1.2 - Система FitBark

1.1.3 Порівняння обраних систем моніторингу

Для наочного порівняння основні параметри обраних систем приведено в порівняльній таблиці 1.1.

Таблиця 1.1 - Порівняння параметрів існуючих рішень з розробленою системою

Порівнювані параметри	Назва системи		
	PetPace Smart Collar	FitBark	PHT(Pet Health Tracker)
Розташування	+	+	+

Порівнювані параметри	Назва системи		
	PetPace Smart Collar	FitBark	PHT(Pet Health Tracker)
Температура	+	-	+
Пульс	+	-	+
Контроль дихальної системи	+	-	-
Контроль сну	-	+	-
Заряд батереї	6 тижнів	14 днів	не потребує заряджання

Згідно таблиці можна зробити висновок, що розроблювана система відслідковує основні життєві ознаки тварини. Проте суттєвою перевагою є її автономність - система не потребує заряду.

1.2 Огляд та аналіз існуючих додатків

1.2.1 Додаток PetPace Smart Collar

Мобільний додаток PetPace, який зображено на рисунку 1.3 дозволяє відстежувати життєвий стан домашньої тварини - температуру, пульс, дихання, сон, активність, позиції, калорії та інше.

Також програма можете завантажити звіти про стан здоров’я, аналізуючи тенденції здоров’я, отримувати сповіщення в режимі реального часу, якщо виявлені ненормальні параметри.

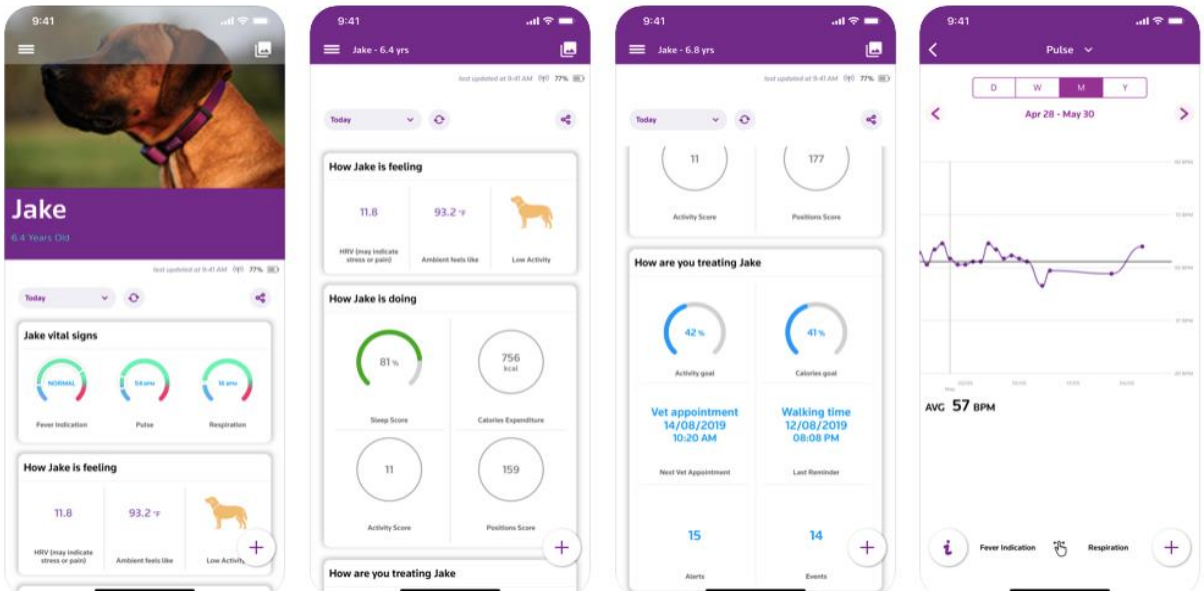

Рисунок 1.3 - Додаток PetPace Smart Collar

Особливості додатку PetPace Smart Collar:

– постійний моніторинг - цілодобовий моніторинг життєвих знаків вашого улюбленця за допомогою PetPace (недоступний у безкоштовній версії);

– оповіщення про здоров'я - сповіщення, коли виявлено відхилення від нормальної поведінки;

– діаграми - перегляд біометричних даних за допомогою різноманітних графіків;

– оцінка активності та регулярності діяльності;

– щоденні цілі;

– щоденник життя - фіксація подій в житті тварини (наприклад візити до ветеринара).

1.2.2 Додаток FitBark

Мобільний додаток FitBark, який зображено на рисунку 1.4 дає доступ до цілодобового моніторингу активності та сну домашньої тварини. Завдяки заздалегіть створеним в додатку безпечним зонам навколо доданих Wi-Fi мереж, господар може відслідковувати розташування розташування домашнього улюбленця. У додатку є можливість налаштувати отримання сповіщень, коли тварина залишає безпечну зону.

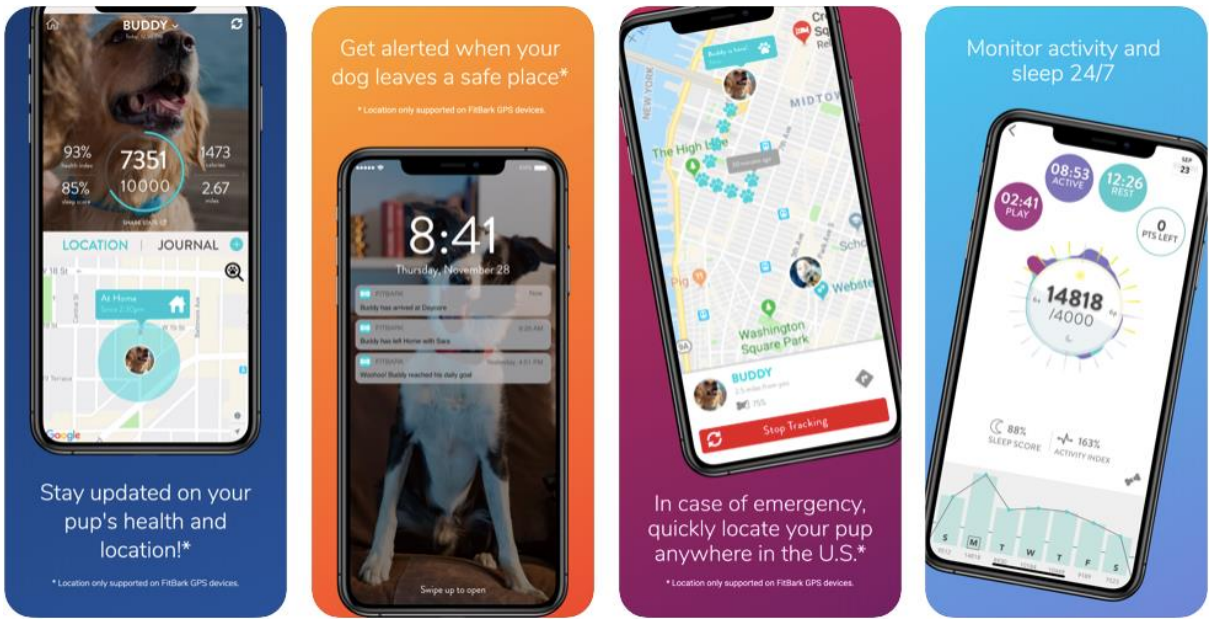

Рисунок 1.4 - Додаток FitBark

Для мешканців Сполучених Штатів Америки доступна функція “надзвичайної ситуації”, за допомогою стільникової служби, господар за 1 хвилину отримає оновлення даних.

Особливістю додатку є можливість поєднати дані власника та тварини в одному додатку, ставити сімейні цілі і ділитись інформацією, через внутрішню соціальну мережу.

1.2.3 Порівняння обраних додатків

Для розгляду існуючих рішень було взято два додатки згідно обраним системам моніторингу: PetPace Smart Collar та FitBark. В порівняльній таблиці 1.2 виведені основні параметри, якими вони володіють.

Таблиця 1.2 - Порівняння параметрів розглянутих додатків

Порівнювані параметри	Назва додатку		
	PetPace Smart Collar	FitBark	PHT(Pet Health Tracker)
Моніторинг стану домашньої тварини	+	+	+
Діаграми біометричних даних	+	-	+
Планування цілей	+	+	-
Показники біометричних даних	+	-	+
Показники геопозиції	+	+	+
Множина тварин	+	+	+
Тип тварини	Собака	Собака	Собака Кіт Корова Коза Вівця Кінь

Згідно таблиці можна зробити висновок, що розроблюваний додаток має основний функціонал, проте відрізняється серед аналогів можливістю застосування його для різних типів домашніх тварин.

Висновки до розділу

У цьому розділі було розглянуто доступні системи моніторингу за станом домашніх тварин та розроблені до них додатки. Було виділено основні особливості кожної системи та створено порівняльну таблицю для наглядної оцінки переваг та недоліків кожної. Для розробки майбутньої системи моніторингу стану домашніх тварин відібрано найважливіші параметри існуючих рішень, з метою створити удосконалену систему, яка не буде мати недоліків всіх існуючих рішень

2 РОЗРОБКА СХЕМИ ЕЛЕКТРИЧНОЇ СТРУКТУРНОЇ

Головна підстава для створення структурної схеми - наглядне зображення алгоритму роботи системи і розуміння підстави для її розробки. Структурна схема відображає основні блоки(основні частини схеми - пристрої, елементи), вузли та головні зв'язки між ними. Побудована схема має відображати логічність і послідовність взаємодії між блоками системи.

Структурна схема у відповідності з наведеними вихідними даними представлена на кресленику ІА61.100БАК.005Э.1. На схемі електричній структурній зображені датчики системи в поєднанні з обчислювальним пристроєм представленим мікроконтролером (МК). До складу датчиків входять: акселерометр, гіроскоп, датчик пульса, цифровий датчик температури, GSM модем. Також на схемі наявна термоелектрична батарея, яка живить систему.

Окремо слід розглянути запропоновані в схемі пристрої:

а) термоелектрична батарея - це пристрій, що використовується для вимірювання теплового випромінювання, що складається з декількох термопар, з'єднаних послідовно. Кожне друге з'єднання зачерняється для поглинання теплового випромінювання, інші екранують випромінювання. Термоелектрична батарея зберігає енергію при зарядці шляху перетворення тепла в енергію і виробляє електрику, коли розряджається.В основі термоелектричної генерації лежить ефект Зеєбека, який представляє собою процес появи різниці потенціалів між двома з'єднаними пластинами з різних матеріалів внаслідок нагрівання зазначеної області. Ефект полягає в створенні термопари, яка складається з двох різнорідних металів, що утворюють один з одним замкнутий контур. Метали один від одного відрізняються різними коефіцієнтами Зеебека, внаслідок

чого виникає напруга між нагрітим провідником термопари і ненагрітими провідником. Ця напруга прямо пропорційно різниці їх температурних значень. Конструкція термоелекричної батареї зображена на рисунку 2.1.

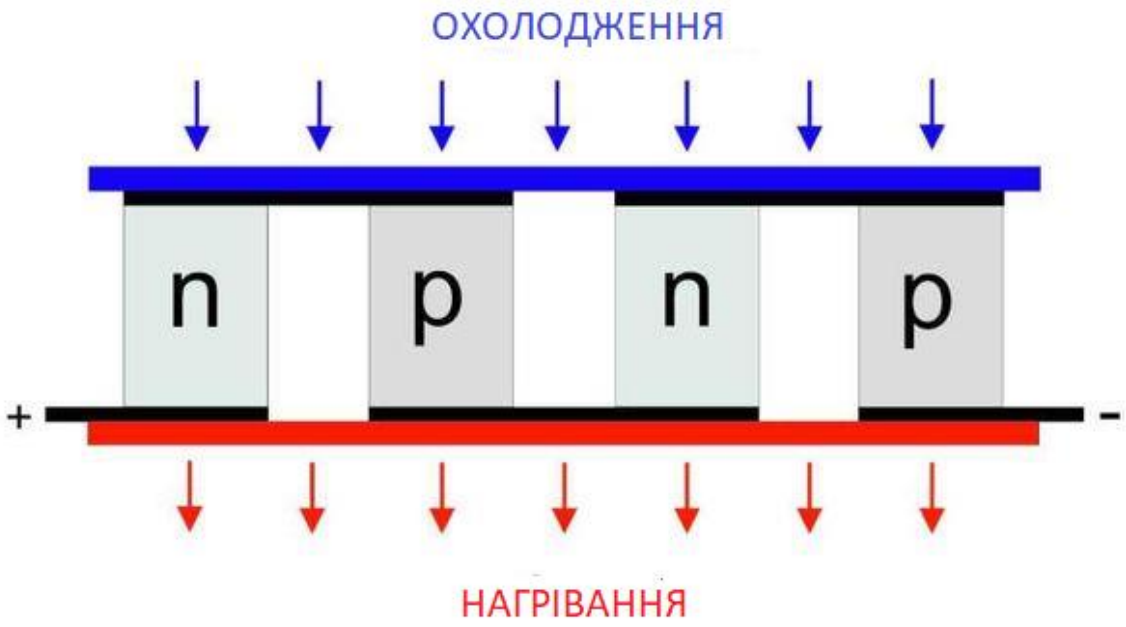

Рисунок 2.1 - Конструкція термоелектричної батареї

У багатьох термоелектричних пристроях застосовується ефект Зеєбека. У більшості випадків в структуру термоелектричних генераторів включаються термобатареї, які набираються з напівпровідникових термічних елементів. При включенні джерела електричного току і під його дією в термоелементах виникає різниця температур його контактів, пропорційна наданій електричній потужності.Зазвичай такі батареї використовуються у колекторах бортових джерел електроживлення космічних приладів. У минулому подібні пристрої застосовувались у навігаційних маяках, радіомаяках, метеостанції та подібному обладнанні, встановлених у місцях,

де технічно чи економічно, можливо, використовуються інші джерельні електролічильники. Насьогодні, такий тип батареї знайшов своє застосування і в повсякденному житті людей. Їх почали використовувати для створення смарт-годинників, які заряджаються від температури тіла. Тобто, чим більше теплової енергії при фізичному навантаженні виробляє тіло, тим більше електрики накопичує пристрій з використанням подібної батареї. У таблиці 2.1 приведене сумарне метаболічне тепловиділення тварин в залежності від маси тіла.

Таблиця 2.1 - Середня сумарна потужність тепловиділення тварин

Маса	Потужність
50 кг	60 Вт
100 кг	115 Вт
150 кг	150 Вт
200 кг	190 Вт

Згідно з дослідженнями, людський організм в середньому виділяє близько 100 Вт тепла, але при інтенсивному навантаженні показник може зрости - до 1 кВт. Тож для прикладу, цієї енергії достатньо для закипання 10 літрів води.В таблиці 2.2 приведена залежність роботи температури від стану тіла та температури середовища.

Таблиця 2.2 – Дієздатність батареї в залежності від умов

Стан тіла	Температура середовища		
	15°C	21°C	26°C
В стані спокою	Заряджається швидко	Заряджається повільно	Не заряджається
Пересування	Заряджається дуже швидко	Заряджається швидко	Заряджається повільно
Біг	Заряджається дуже швидко	Заряджається дуже швидко	Заряджається швидко

b) Датчик температури - пристрій, що реагує на температуру шляхом видачі електричного сигналу або спрацьовує механічно.Пристрій дозволяє виміряти температуру об'єкта або речовини, використовуючи при цьому різні властивості і характеристики вимірюваних тіл або середовища. Цифрові датчики можуть підключатися через досить довгі провідні лінії; на відміну від слабких аналогових сигналів, які надходять з інших типів датчиків, цифровий сигнал стійкий до впливу перешкод. Кожен вид домашніх тварин має свою сталу температуру тіла яка лежить в деякому діапазоні, в якому метаболізм функціонує без змін (термонейтральна зона). Відхилення поза цим діапазоном, який порівняно вузький призводить до збільшення метаболізму, що свідчить про проблеми або відхилення від норми у функціонуванні організму.Діапазон температур тіла здорової тварини згідно її виду приведено в таблиці 2.1.

Таблиця 2.1. - Сталі діапазони температури тварин

Тип тварин	Діапазон температур
Собака	37 ,5—39,0
Корова	38,2—39.5
Кінь	37,5—38,5
Коза	39,0—40,5
Вівця	39,0—40,5
Кіт	38,0—39,0

c) Датчик серцевого ритму - пристрій персонального моніторингу частоти скорочень серця в реальному часі. Для розробки було обрано оптичний датчик пульсу, який за допомогою світлодіодів просвічує шкірний покрив потужним пучком світла. Потім відбувається вимір відбитого кількості світла, розсіяного кровотоком. Технологія будується на тому, що розсіювання світла в тканинах відбувається певним чином в залежності від динаміки кровотоку в капілярах, що дозволяє відстежити зміни пульсу.Варіація частоти серцевих скорочень зазвичай відображає стрес, передчуття, рух, навантаження, і різні захворювання, що є досить важливими показниками при моніторингу стану домашньої тварини. Частота серцевих скорочень - найважливіший параметр здоров’я. Загалом для кожної категорії тварин визначено свої діапазони, які визначають задовільний стан, що приведено в таблиці 2.2.

Таблиця 2.1. - Сталі діапазони пульсу тварин

Тип тварин	Діапазон пульсу
Собака	70-120
Корова	24-44
Кінь	50-80
Коза	70-80
Вівця	70-80
Кіт	110-120

d) Гіроскоп - навігаційний пристрій, здатний реагувати на зміну кутів орієнтації тіла, на якому воно встановлено, щодо інерціальної системи відліку. Він фіксує положення об'єкта в просторі відносно трьох площин.

e) Акселерометр - прилад, призначений для виміру проекції уявного прискорення. Він може застосовується для вимірювання проекцій абсолютного лінійного прискорення (якщо відомі величина і напрямок гравітаційного прискорення в даній точці простору), що знайшло застосування в створенні інерційних навігаційних систем, де отримані за допомогою акселерометрів вимірювання інтегрують, отримуючи інерційну швидкість і координати носія. Таким чином, акселерометри, нарівні з гіроскопами, є невід'ємними компонентами систем навігації.

f) Мікроконтролер - мікросхема, призначена для управління електронними пристроями. У типовому мікроконтролері є функції процесора, периферійних пристроїв, а також міститься оперативна пам'ять і / або ПЗП (постійний запам'ятовуючий пристрій).

g) Мікроконтролер використовується для управління роботою системи моніторингу. Дані з обраних вище датчиків передаються на МК, який обробляє отриманий сигнал і передає дані далі для обробки та зберігання в БД.

Висновки до розділу

У даному розділі було розроблену схему електричну структурну, яка продемонструвала сутність розроблюваної системи. Додатково було описано основні блоки та їх значення згідно описуємої системи. Для розробки обрані найсучасніші прилади та рішення, які можуть оптимізувати та полегшити роботу систему.

3 РОЗРОБКА СХЕМИ ЕЛЕКТРИЧНОЇ ПРИНЦИПОВОЇ

Принципова електрична схема - графічне зображення (модель), що служить для передачі за допомогою умовних графічних і буквено-цифрових позначень (піктограм) зв'язків між елементами електричного пристрою.

Принципова схема, на відміну від розведення друкованої плати не показує взаємного (фізичного) розташування елементів, а лише вказує на те, які висновки реальних елементів (наприклад, мікросхем) з якими з'єднуються. Ця схема дає детальне уявлення про роботу системи і служить для вивчення принципу дії системи, вони необхідні при виробництві системи і в її експлуатації.

Принципова електрична схема у відповідності з наведеними вихідними даними представлена на кресленику ІА61.100БАК.005Э3. Система з одного контуру, в якому обрані датчики збирають інформацію та передають дані до мікроконтролера. Для розробки частково були взяті готові модулі, які допомогли прискорити та полегшити роботу. Самі по собі готові модулі містять всередині окремі схеми, проте для поточної розробки використані їх укомплектовані мікросхеми. Окремі схеми датчиків містять в собі споміжні елементи, які узгоджують роботу ланцюга та виконують запобіжну функцію. Так в розробці були використані схеми з використанням елементів MPU6050, ATMEGA32, DS18B20, SIM900D.

3.1 Вибір окремих елементів

Розвиток сучасних технологій дає можливість здійснити автоматизацію процесів управління та збір інформації за допомогою розробки схеми з використанням готових рішень. Для виконання задачі необхідно обрати контролер, акселерометр і гіроскоп, датчик температури і GSM модуль.

3.1.1 Акселерометр та гіроскоп

Мікросхема MPU-6050 містить два трьох осьові пристрої: акселерометр і гіроскоп, що зображено на рисунку 3.1.

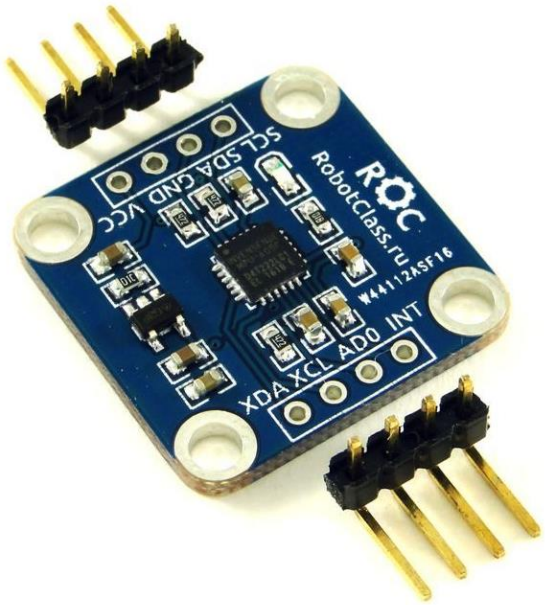

Рисунок 3.1 - Акселерометр та гіроскоп MPU-6050

Їх дані проходять попередню обробку і передаються по послідовному інтерфейсу I2C в мікроконтролер. Даний датчик відмінно підходить для визначення положення в просторі.

Характеристики MPU-6050:

- протокол: IIC (він же I2C, він же TWI);
- живлення: 3 - 5В;
- діапазон гіроскопа: + 250 500 1000 2000 ° / с;
- вбудований 16-бітний АЦП;
- діапазон акселерометра: ± 2 ± 4 ± 8 ± 16 g.

3.1.2 Мікроконтролер

Atmega32 - 8-ми бітовий мікроконтролером побудований на розширеній AVR RISC архітектурі. Завдяки командам, які виконуються за один машинний такт, контролер досягає продуктивності в 1 MIPS на робочій частоті 1 МГц, що дозволяє розробнику ефективно оптимізувати споживання енергії за рахунок вибору оптимальної продуктивності. Мікроконтролер Atmega32 приведено на рисунку 3.2.

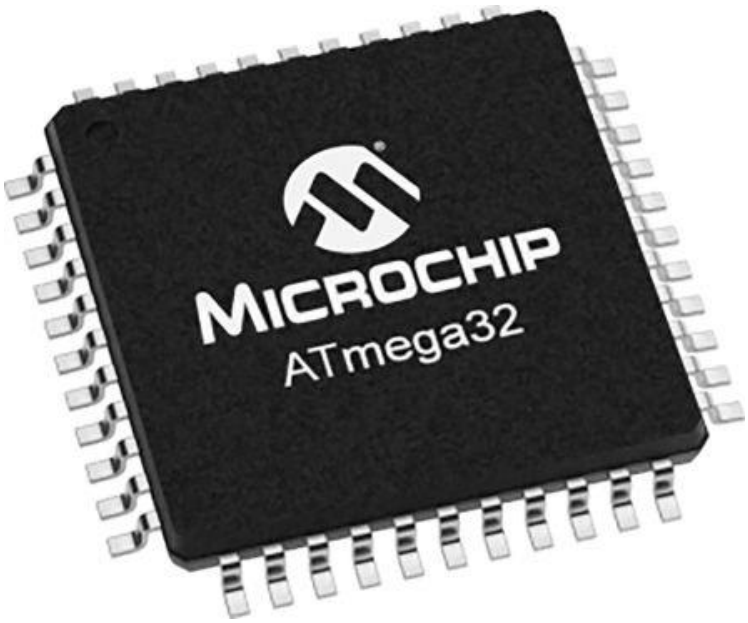

Рисунок 3.2 – Мікроконтролер Microchip Technology ATMEGA32-16MU

Характеристики Atmega32:

- тактова частота: 16 МГц;
- розмір програмної пам'яті: 32 кБ;
- пам'ять даних (EEPROM) 1024 Б;
- пам'ять ОЗУ (SRAM) (RAM): 2048 Б;
- кварцовий резонатор: 32кГц;

- кількість каналів аналого-цифрового перетворювача (АЦП): 8x10-bit;
- кількість ліній вводу-виводу (I / O): 32;
- типи послідовних інтерфейсів і кількість каналів: UART; SPI; I2C;
- напруга живлення: 4.5 - 5.5 В.

3.1.3 Цифровий датчик температури

DS18B20 - це цифровий вимірювач температури, з дозволом перетворення 9 - 12 розрядів і функцією тривожного сигналу контролю за температурою, що зображений на рисунку 3.3. Параметри контролю можуть бути задані користувачем і збережені в енергонезалежній пам'яті датчика. DS18B20 - це цілий мікроконтролер, який може сигналізувати про вихід температури за встановлені межі, міняти точність вимірювань, спосіб взаємодії з контролером і багато іншого.

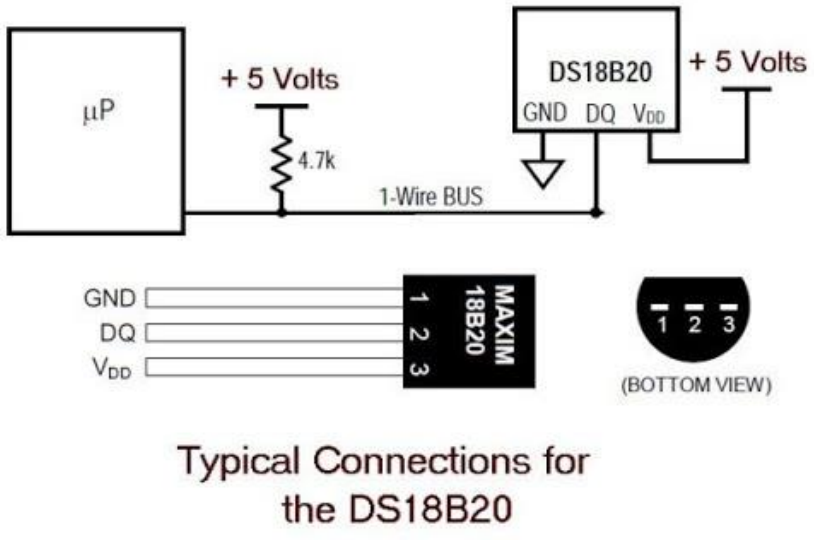

Рисунок 3.3 – Схема підключення цифрового датчика температури DS18B20

Характеристики DS18B20:

- напруга живлення: 3.0 - 5.5 В;
- діапазон вимірюємих температур: -55°C - +125°C;
- точність температури: 0.5 °C.

3.1.4 GSM модуль

SIM900D - широко поширений GSM / GPRS модуль, який може бути використаний навіть в аматорських пристроях і дозволяє уникнути складних технологій пайки і монтажу. Є вбудованими TCP / IP стек. Модуль в поєднанні з SIM CARD 1,8/3V виконує функцію GPS-трекера.

Описана вище модель GSM модуля зображено на рисунку 3.4

Рисунок 3.4 - GSM модуль SIM900D

Характеристики SIM900D:

- піапазони: GSM 850/900/1800/1900 МГц;
- клас потужності 4 (2 Вт в діапазонах 850/900 МГц);
- клас потужності 1 (1 Вт в діапазонах 1800 / 1900MHz);
- вбудований стек TCP / IP;
- напруга живлення 3,2 - 4,8 В;
- робочий температурний діапазон: -30 ° С ... +80 ° С.

У даній схемі GSM модем використовується для передачі інформації про стан тварини, зібраних за допомогою датчиків. Його робота схожа на роботу мобільного телефону, який потребує SIM-карту для коректності своєї роботи. GPS модем отримує сигнали зі супутника і розраховує широту і довготу розташування тварини, після чого відправляє дані до контролера у вигляді послідовності даних.

Висновки до розділу

У даному розділі було розроблену схему електричну принципіальну, яка продемонструвала зв’язки між елементами електричних пристроїв розроблюваної системи. Додатково було описано основні модулі використані в розробці та розкрито їх зміст, приведено характеристику кожного з них. Для розробки були обрані найоптимальніші рішення, які самі по собі мають перераховану технічну характеристику

4 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ДОДАТКУ

Ідея розробленого додатку полягає в створенні сучасної платформи для збору та обробки даних з системи моніторингу стану домашніх тварин. Програма дає можливість користувачу створити профілі всіх своїх домашніх улюбленців та відслідковувати основні життєві показники в зручному форматі. В додатку є наступні модулі:

1) Activity - модуль, в якому користувач може ознайомитись з історією переміщення тварини за окремий період часу. Робота модуля Activity зображена на рисунку 4.1.

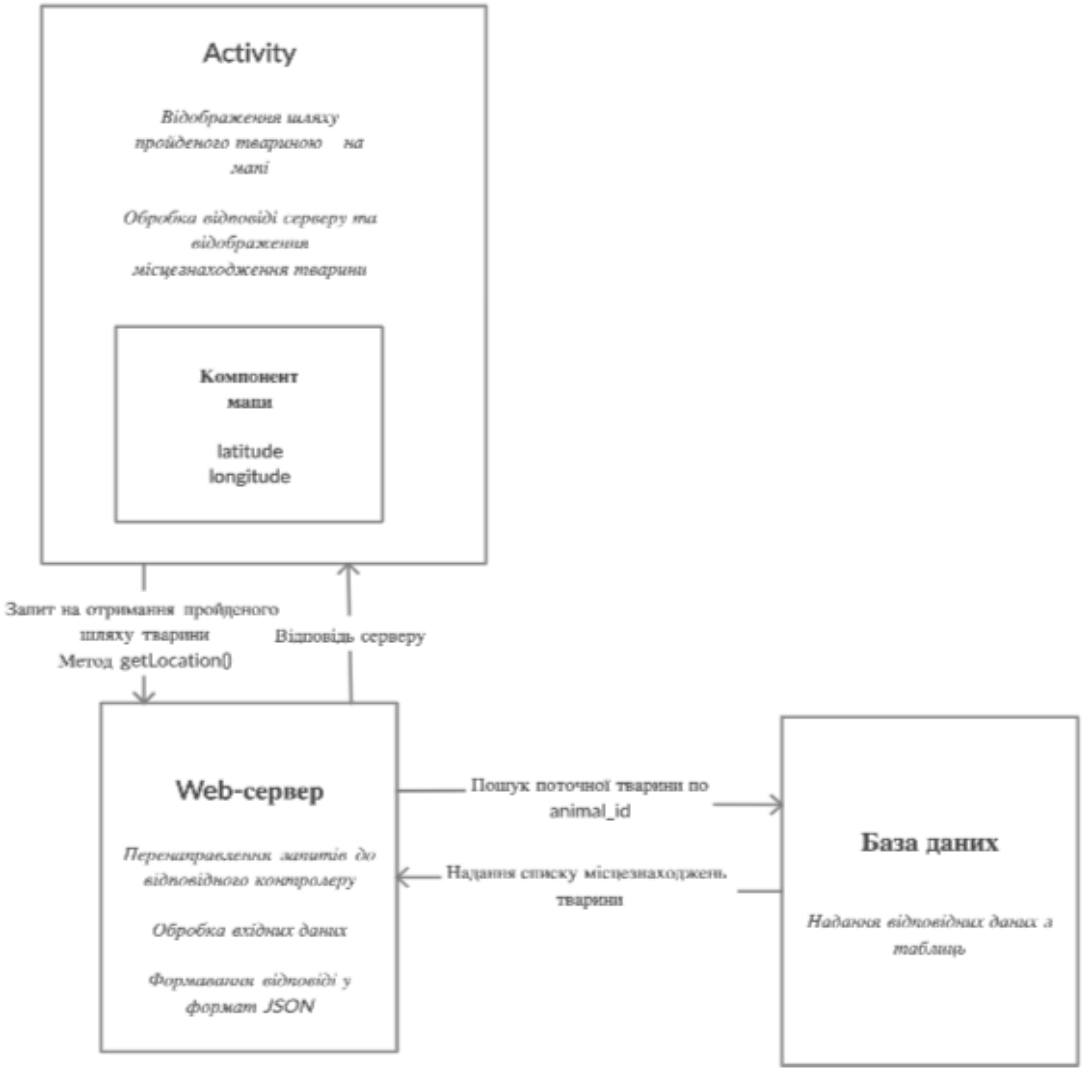

Рисунок 4.1 – Схема роботи модуля Activity

2) Health - модуль, в якому користувач може ознайомитись з поточним станом тварини(пульс та температура) та переглянути графіки зміни показників за окремий період часу. Також в додатку продумане попередження господаря про відхилення даних від норми, згідно типу тварини. Робота модуля Health зображена на рисунку 4.2.

Рисунок 4.2 - Робота модуля Health

3) Animals - модуль, в якому користувач може створювати та редагувати дані про своїх домашніх улюбленців(ім’я, стать, тип та вік). Робота модуля Animals зображена на рисунку 4.3.

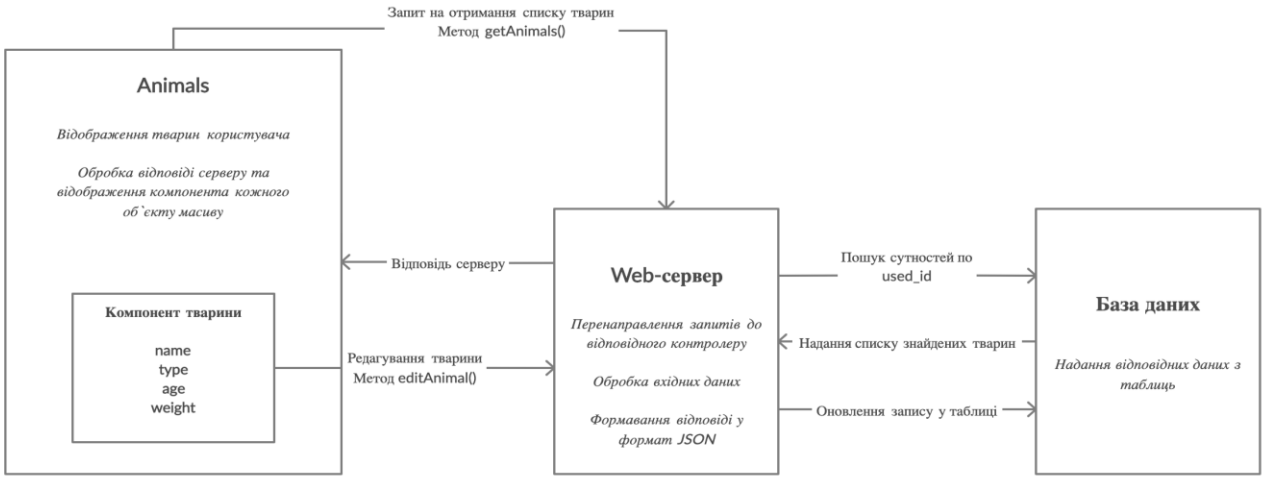

Рисунок 4.3 - Робота модуля Animals

4) Live GPS - модуль, в якому користувач може спостерігати за поточним розташуванням тварини відносно заданих безпечних зон. Робота модуля Live GPS зображена на рисунку 4.4.

Рисунок 4.4 - Робота модуля Live GPS

5) Profile - модуль, в якому користувач може задати безпечні точки на карті(дім), та змінити пароль.

4.1 Мова програмування та платформа

Розроблюваний додаток створено використовуючи об’єктно-орієнтовану мову програмування - JavaScript (JS). Ця мова найчастіше застосовується для розробки веб-сторінок, адже з її допомогою можлива взаємодія з користувачем, асинхронний обмін даними з серверами, зміна структури та зовнішнього вигляду веб-сторінок на стороні клієнта. JavaScript є скриптовою мовою програмування з динамічною типізацією, що робить її простою та більш зручною в розробці.

В даному випадку мова використовується для програмування на стороні сервера Node.js.

Node.js – програмна платформа з відкритим кодом, заснована на V8, транслюючий JavaScript в машинний код, що перетворює JavaScript в мову

загального призначення. Платформа виконує роль веб-сервера. Node.js дає можливість підключати сторонні бібліотеки, розроблені на різних мовах програмування, дозволяє JavaScript взаємодіяти з пристроями введення-виведення через свій API.

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів (callback).

4.1.1 Бібліотеки використані для back-end розробки

Для розробки додатку були використані наступні збірники підпрограм:

1) Express.js - це мінімалістичний і гнучкий web-фреймворк для додатків Node.js, що надає великий набір функцій для мобільних і веб-додатків. Express, реалізований як вільне і відкрите програмне забезпечення, проєктований для створення веб-додатків і API. Вважається стандартним каркасом для Node.js.
2) Passport.js - це проміжне програмне забезпечення для Node.js. Через свою неймовірну гнучкість і модульність, Passport можна використовувати в будь-якому веб-додатку на основі Express.js. Passport - це middleware для перевірки автентичності, яку в додатку використано для управління сесіями.
3) Sequelize - це ORM-бібліотека(Object-Relational Mapping) для додатків на Node.js, яка здійснює зіставлення таблиць в базі даних і відносин між ними з класами. При використанні Sequelize ми можна працювати з даними як зі звичайними об'єктами. В даному випадку Sequelize працює СУБД - MySQL.

4.2 Проєктування додатку

Для оптимального розв’язання поставленого завдання при проєктуванні були використані породжуючі, структурні та архітектурні шаблони.

a) Creational - шаблони проєктування, які абстрагують процес інстанціювання. Шаблон, який породжує класи, використовує спадкування, щоб змінювати інстанціюємий клас, а шаблон, який породжує об'єкти, делегує інстанціювання іншому об'єкту.
    1) Abstract factory - клас, який представляє собою інтерфейс для створення компонентів системи.
b) Structural - шаблони, визначають різні складні структури, які змінюють інтерфейс вже існуючих об'єктів або його реалізацію, дозволяючи полегшити розробку і оптимізувати програму.
    1) Proxy - об'єкт, який є посередником між двома іншими об'єктами, і який реалізує / обмежує доступ до об'єкта, до якого звертаються через нього.
    2) Composite - об'єкт, який об'єднує в собі об'єкти, подібні до нього самого.
c) Архітектурний шаблон висловлює фундаментальну структурну схему організації системи програмного забезпечення. Він відображає набір виділених підсистем, позначає їх ролі і включає правила і рекомендації з організації зв'язків між ними.
    1) Model-ViewViewModel (MVVM) - це патерн розробки, що дозволяє розділити додаток на три функціональні частини:
        – Model - основна логіка програми (робота з даними, обчислення, запити і т.д.).
        – View - вид (призначений для користувача інтерфейс).
        – ViewModel - модель уявлення, яка служить прошарком між View і Model.

Такий поділ дозволяє прискорити розробку і підтримувані програми - можна міняти один компонент, не зачіпаючи код іншого.

4.3 Опис створеної бази даних

База даних - це організована структура, призначена для зберігання, зміни та обробки взаємозв'язкової інформації, що має більші об'єкти.

Дані, що містяться в найпоширеныших типах сучасних БД, зазвичай формуються у вигляді стовпців та рядків у ряді таблиць, щоб забезпечити ефективну обробку даних. Це дозволяє легко отримати доступ до даних, керувати ними, змінювати, оновлювати, контролювати та застосовувати їх. У великій базі даних для записів і запитів даних використовується мова створених запитів (SQL).

Система управління базами даних (СУБД) - це комплексний програмного забезпечення, необхідного для створення структури нової бази, її використання, редагування вмісту та відображення інформації.

В даній роботі була використана MySQL - найпопулярніша база даних з відкритим кодом у світі. ЇЇ можна охарактеризувати, як мову структурованих запитів. MySQL є дуже швидкою, надійною і легкою у використанні, тому саме вона була обрана для реалізації БД в дипломному проекті. В таблиці 4.1 перераховані всі наявні в створенні БД таблиці та їх параметри.

Таблиця 4.1 - Таблиці БД та їх параметри

Назва таблиці	Параметри
Temperature	- temperature; - animal_id.

Назва таблиці	Параметри
Animal	- name; - animal_type; - age; - status; - temperature; - pulse; - location; - user_id.
Location	- lat; - lang; - location; - animal_id.
Pulse	- pulse; - heart_status; - animal_id.
Authentication	- username; - password; - token.

Назва таблиці	Параметри
Heart_Satus	- stable; - high; - critical; lowered.
Authentication	- username; - password; - token.
User	- name; - password; - collar_status.

База даних розроблена по принципу роботи класів, тобто звернення до таблиці відбувається як до простого об’єкту, а пошук та сортування даних відбувається по унікальному ідентифікатору.

4.4 Опис серверної частини системи

Веб-сервер - сервер (програма), що приймає HTTP-запити від клієнтів, зазвичай веб-браузерів, і видає відповідь у вигляді HTML сторінок, на яких може міститися різна інформація: зображення, тексти, скрипти, файли, медіадані (відео та аудіо) і багато іншого.

Структура представляє собою базову взаємодію усього функціоналу серверу, яка поділяється на router, controller та model, що представлено на рисунку 4.5.

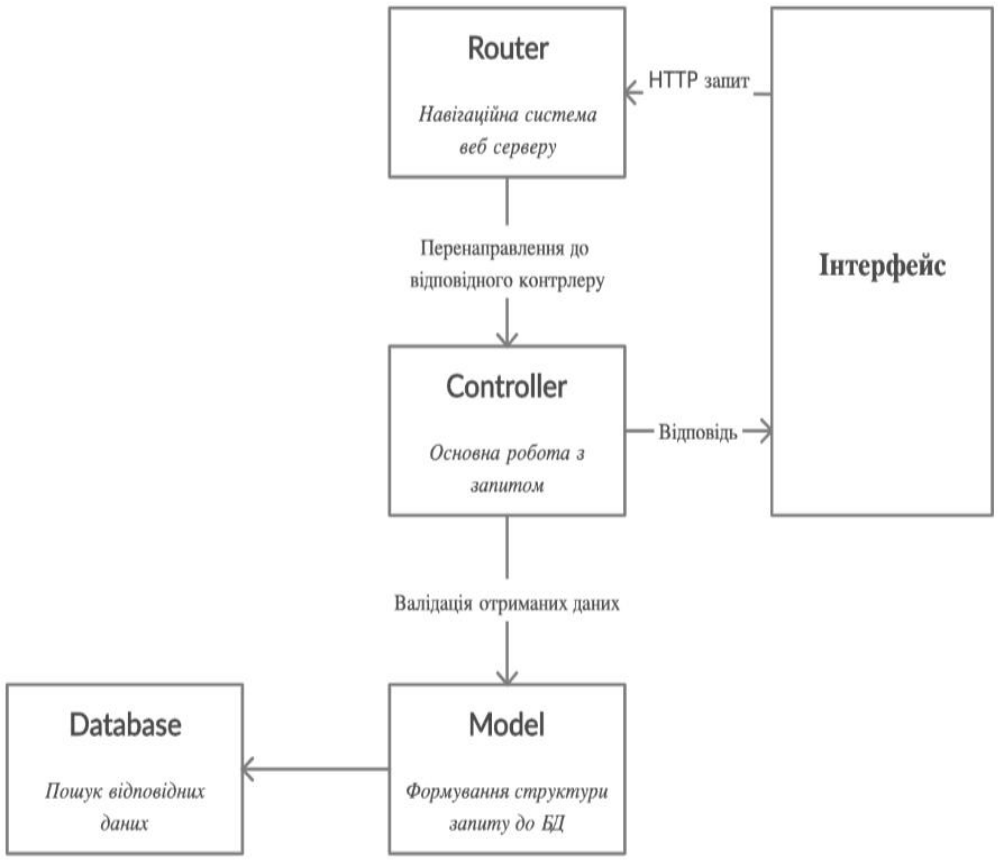

Рисунок 4.5 - Структура роботи веб-серверу

Router - відповідальний модуль за навігацию HTTP запитів до відповідального контролеру та перевірку правильного формування вхідних даних до методу контроллера. Якщо перевірка невдала то сервер видає помилку на запит. Код відповіді (стану) HTTP показує, чи був успішно виконаний певний HTTP запит.

Controller - основний функціонал серверу, відповідальний за обробку та фільтрацію даних як з запиту та з бази даних. Створює новий запит до бази даних та звертається до моделі сутності.

Model - задана структура сутності завдяки якій сервер спілкується з базою даних та формує правильну структуру даних.

Приклад роботи HTTP протоколу з веб-сервером зображено на рисунку 4.6

Рисунок 4.6 - Аутентифікація користувача

На прикладі зображено аутентифікацію користувача в систему з генерацією JWT токена.

Веб-токен JSON, або JWT - стандартизований, в деяких випадках підписаний і/або зашифрований формат упаковки даних, який використовується для безпечної передачі інформації між двома сторонами.

4.5 Розробка модуля відстежування

Модуль відстеження в додатку займає одну з найцікавіших позицій. Він реалізований таким чином, що у користувача є можливість не лише просто слідкувати за місцезнаходженням тварини, але і за її переміщенням.

Для розробки модуля, основним питання було створення рекурсивного модуля.

Модуль повинен виконувати такий функціонал:

- створення GET запитів до БД;

- оновлення та структурування даних;
- відправка інформації до інтерфейсу користувача;

Рекурсивний метод – це повторюване виконання певної функції в методі. При створенні даного методу був використаний метод асинхронних функцій. Рекурсивний метод зображено на рисунок 4.7

```
const express = require("express");
const app = express();
const port = 3000;
const unirest = require("unirest");

app.get("/", (req, res) => {
  var apiCall = unirest("GET",
    "https://ip-geolocation-ipwhois-io.p.rapidapi.com/json/"
  );

  apiCall.headers({
    "x-rapidapi-host": "ip-geolocation-ipwhois-io.p.rapidapi.com",
    "x-rapidapi-key": "srclZqaa9imshAk9Xzz55u27oltLp1SqdiFjsnmva9PTpf2j3f"
  });

  apiCall.end(function(result) {
    if (res.error) throw new Error(result.error);
    console.log(result.body);
    res.send(result.body);
  });ї
});

app.listen(port);
```

Рисунок 4.7 – Налагодження методу

При розробці методу була використана бібліотека RapidApi для тестування та налаштування парсингу даних. Переваги використаної бібліотеки:

- поля X-RapidAPI-Host та X-RapidAPI-Key вже заповнені для використовування.
- необов'язкове поле IP-адреси. Якщо поле IP-адреси заповнене то запит оброблятиметься швидше, або IP-адреса поточного

користувача буде автоматично виявлена.

Запит надсилається кожні 10 секунд, контролер звертається до моделі Location, яка в свою чергу отримує оновленні дані про місцезнаходження. Отримані дані надаються сховищу інтерфейсу додатку.

4.6 Опис розробленого інтерфейсу системи

Важливим питанням створення інтерфейсу було динамічне оновлення показників тварин , через це була обрана бібліотека яка повністю вирішує поставлене питання – Vue.js.

Vue.js - це стрімко розвиваюча інфраструктура front-end для JavaScript, натхненна Angular.js, Reactive.js і Rivets.js, яка пропонує спрощений дизайн користувальницького інтерфейсу, маніпулювання і глибоку реактивність. Оскільки Vue працює тільки на «рівні уявлення» і не використовується для проміжного програмного забезпечення і бекенда.

Завдяки схожій на MVVM структурі, яка заснована на концепції двосторонньої прив'язки даних до компонентів і уявленням, фреймворк перевершує по швидкості інші систем JS верхнього рівня і робить його дуже зручним в інтегруванні та прототипуванні.

Основною перевагою цього фреймворку є архітектура роботи реактиновті.

Реактивність – це можливість динамічного оновлювання стану представлення інтерфейсу додатку. Кожен vue компонент складається з екземпляру vue та HTML структури.

Екземляр vue – це шаблон поєднання структури сторінки та даних функціоналу інтерфейсу. Поєднання відбувається з допомогою декларатив фреймворку:

- v-model – строкова команда зв’язку даних інтерфейсу та необхідного тег сторінки;
- v-click – подія ініціалізації кліку на необхідний тег сторінки;

- v-if – умовне відображення структури сторінки в залежності від вхідних даних;
- v-for – реалізація циклу for для структури сторінки.

Використання основних декларатив та взаємодія функціоналу інтерфейсу з структурою сторінки представлено на рисунку 4.7

```
<section class="text-black">
  <q-table
    v-if="data"
    :columns="columns"
    :data="data"
    row-key="id"
    color="black"
    separator="cell"
    dense
  >
    <template v-slot:top>
      <section dense class="row justify-between items-center full-width">
        <div class="col q-table__title">
          {{ $t('Animals') }}
        </div>

        <div class="col-auto">
          <q-btn
            round
            color="green-4"
            size="sm"
            icon="add"
            @click="$emit('create')"
          />
        </div>
      </section>
    </template>
```

Рисунок 4.7 – Взаємодія інтерфейсу додатку з структурою сторінки

Отже можна зробити висновок, що перевагою цієї бібліотеки над іншими є простота взаємодії між HTML та функціоналом інтерфейсу.

Унікальністю фреймворку є - сховище, яке представлене у вигляді шаблону управління станом в поєднанні з бібліотекою для додатків Vue.js. Це служить централізованим сховищем для всіх компонентів програми, з правилами, що гарантує тільки передбачувані зміни стану. Він також інтегрується з офіційним розширенням інструментальних засобів Vue для надання додаткових функцій, таких як налагодження з нульовим часом

відгуку і експорт / імпорт моментальних знімків стану. Використовуючи сховище даних централізованого додатку, весь стан програми може бути представлено в одному місці, що робить додаток більш організованим.

Приклад структури сховища розробленого додатку наведено на рисунку 4.7.

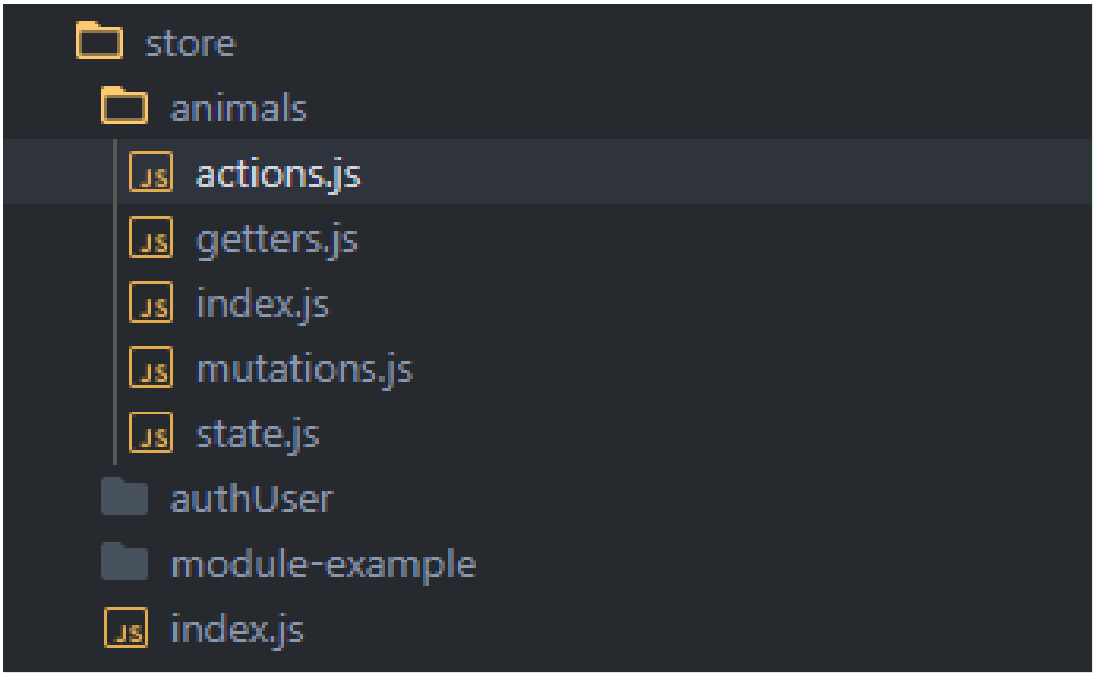

Рисунок 4.7 - Реалізація сховища

Структура сховища має 4 основних методу спілкування та обробки даних:

1) Getters - надає доступ до сховища та спілкування.

2) Mutations - оновлює стан сховища та змінних.

3) Actions - реалізує основну функцію роботи сховища

4) State - зберігає всі змінні сховища.

Окремо варто розглянути питання компонентної розробки веб-додатку. Компонентні фреймворки дозволяють швидко будувати додатки, використовуючи готові будівельні блоки - компоненти. Вони дають

можливість створювати додатки як малої, так і середньої складності.

4.7 Компоненти

Компоненти є основним засобом розширення функціональних можливостей програми. Вони призначені для багаторазового використання, мають певний інтерфейс і взаємодіють з програмним середовищем за допомогою подій.

Для спрощення компонентної розробки, компоненти мають бути:

- придатні для повторного використання;
- легкозамінні іншими подібними компонентами;
- незалежні від контексту та інших компонентів;
- розширювані, компонент має змогу розширювати існуючі компоненти для забезпечення нової поведінки;
- інкапсульвоані, мають надавати інтерфейси, що дозволяють викликаючій стороні використовувати їх функціональність, не розкриваючи при цьому деталі внутрішніх процеси.

Основна робота додатки заснована на взаємодії спілкування компонентів між собою. Дуже важливо з самого початку розробки інтерфейсу правильно написати архітектуру взаємодій компонентів програми. Кожен компонент інтерфейсу був створений як незалежний екземляр, такий спосіб надає можливість глобального використовування кожного компоненту а також додавання нового функціоналу не буде впливати на роботу іншого функціоналу.

Для розширення роботи функціоналу існують UI-фреймворки, які надають вже готові компоненти інтерфейсу, для більш комфортної та детальної розробки. Для поточної розробки додатку були обрані бібліотеки

Vuetify та Apexcharts. Для реалізації роботи компонентів була використана бібліотека Vuetify. Бібліотека являє собою вже готові шаблони компонентів для спрощення процесу розробки. Включає в себе готову html верстку і css стилі.Apexcharts – бібліотека надає можливості створення графіків. Кожен параметр відображається у співвідношенні параметр/час, що надає можливість точно аналізувати стан тварини та її активність.

Також розроблена можливість відображати одночас декілька тварин за для порівняння та комфортного відстежування поточного стану тварин. Підключення та використання графіку з використанням реактивності фреймворку Vue. Налагодження графіку представлено на рисунку 4.8.

```
data: function() {
  return {
    chartOptions: {
      plotOptions: {
        bar: {
          horizontal: false,
          endingShape: 'rounded',
          columnWidth: '55%',
        },
      },
      dataLabels: {
        enabled: false
      },
      stroke: {
        show: true,
        width: 2,
        colors: ['transparent']
      },
      xaxis: {
        categories: new Date(),
      },
      yaxis: {
        title: {
          text: '$ (thousands)'
        }
      },
```

Рисунок 4.8. – Налагодження графіку

Основним принципом поєднання реактивності Vue та налагодження компонента графіку є передача форми графіку як об’єкту до вхідних даних екземпляру інтерфейсу. Такий принцип надає можливість динамічного оновлення даних графіку.

Також подібний принцип було використано для реалізації відображення обраної тварини на мапі.

4.8 Опис класів та методів системи

Для кожного з наведеного у першому пункті 4.1 функціоналу був розроблен окремий клас, у якому інкапсулюються методи, необхідні для коректної роботи додатку. Система має 6 класів: Animal, Location, Temperature, Pulse, User та Authentication. Діаграма класів розробленого додатку представлена на кресленику ІА61.100БАК.005.Д1. Діаграма класів визначає типи класів системи і різного роду статичні зв'язки, які існують між ними. На діаграмах класів зображуються також атрибути класів, операції класів та обмеження, які накладаються на зв'язку між класами.

Згідно розробленому кресленику наглядно видно співвідношення між сутностями: так у одного користувача може бути багато тварин(співвідношення між класами User і Animal - один до багатьох), у кожної тварини показники динамічні, тобто вона має різні дані про пульс, температуру і локацію(співвідношення між класами Animal і Location, Temperature, Pulse - один до багатьох), всі користувачі проходять авторизацію, тож співвідношення між класами User та Authentication - один до багатьох.

Детальний опис алгоритму роботи класів і методів розкрито у таблицях 4.2 - 4.7

Таблиця 4.2 - Опис методів класу “User”

Метод	Вхідні параметри	Опис
removeAnimal: Метод відповідає за видалення обраної тварини	- animal_id.	Метод формує запит до таблиці Animals, по параметру animal_id відбувається пошук та видалення знайденого рядка таблиці.
addAnimal: Метод відповідає за додавання нової тварини в профіль користувача.	- name; - age; - type; - temperature; - pulse; - location; - status.	Після заповнення форми, створюється запит до таблиці бази даних Animals для додавання нового запису. Параметри location, pulse, temperature, status в формі користувачеві недоступні, але вони записуються в базу даних як початкові параметри. В подальшому користувач може змінювати статус тварини, точніше нашийника.

Метод	Вхідні параметри	Опис
getAnimals: Метод відповідає за отримання даних про тварин підключених до аккаунту користувача.	- user_id.	Метод звертається до таблиці Animals для сортування Vзаписів і пошуку по id користувача, повертаючи всі записи пов'язані з цим id. Після ініціалізації запиту і отримання даних з таблиці відбувається запит до інших таблиць за допомогою animal_id.
changePassword: Метод відповідає за зміну пароля користувача для забезпечення більшого захисту власного профілю.	- password; - new_password.	Для виводу методу відбувається повторна перевірка user token для перевірки входу користувача у системі. Відбувається звернення до таблиці бази даних для перевірки співпадіння старого пароля. Якщо перевірка пройшла, тоді відбувається зміна старого пароля на новий.

Таблиця 4.3 - Опис методів класу “Animal”

Метод	Вхідні параметри	Опис
editAnimal: Метод відповідає за редагування даних про тварину	- name; - animal_type; - age; - pulse; - location; - temperature; - status.	Метод створює запит до БД, згідно всім отриманим параметрам, який оновлює дані у відповідних рядках таблиці Animals.
setAnimalType: Метод відповідає за визначення типу тварини	- animal_type.	При створенні тварини метод створює запит до БД, яка віддає масив існуючих типів тварин.
setStatus: Метод змінює статус активності тварини відносно трекінгової системи	- animal_id.	Метод створює запит до БД, згідно параметру animal_id, та змінює статус на “true” чи “false”. Ця дія провокує відображення обраної тварини на сторінці LiveGPS.

Таблиця 4.4 - Опис методів класу “Location”

Метод	Вхідні параметри	Опис
getLocation: Метод отримує та передає дані про поточне місце знаходження тварини.	- animal_id.	Метод створює запит до БД, згідно параметру animal_id,
Метод	Вхідні параметри	Опис
setLocation: Метод відповідає за оновлення даних про місцерозположення тварини	- animal_id; - lat; - lng.	Метод визивається кожну хвилину, якщо ошийник тварини активовано, він передає зміну розположення тварини в форматі координат, що дозволяє малювати схему її пересування.

Таблиця 4.5 - Опис методів класу “Temperature”

Метод	Вхідні параметри	Опис
getTemperature: Метод відповідає за встановлення зв’язку між твариною і таблицею Temperature.	- animal_id.	Метод створює запит до БД, згідно параметру animal_id, що передає дані про температуру тіла тварини.

Таблиця 4.6 - Опис методів класу “Pulse”

Метод	Вхідні параметри	Опис
setHeartStatus: Метод визначає стан тварини згідно її пульсу.	- animal_type; pulse.	Метод створює запит до БД, згідно параметру pulse та animal_type, де згідно чотирьом типовим станам відбувається перевірка належності до діапазону стану та повертається статус пульса тварини.
getPulse: Метод відповідає за встановлення зв’язку між твариною і таблицею Pulse.	animal_id.	Метод створює запит до БД, згідно параметру animal_id, що передає дані про пульс тварини.

Таблиця 4.7 - Опис методів класу “Authentication”

Метод	Вхідні параметри	Опис
SignIn: Метод відповідає за вхід користувача в систему.	- username; - password;	Виклик методу створює запит до таблиці Users бази даних і перевіряє правильність введених параметрів користувача, якщо вони вірні викликається метод генерації token користувача для входу в систему.

Метод	Вхідні параметри	Опис
SignOut: Метод відповідає за вихід користувача з активною сесії додатку і видалення всіх його даних з поточної сесії.	- username.	Виклик методу створює запит на видалення user token з локального сховища сесії.
Registration: Метод відповідає за створення нового користувача в системі і додаванні його в таблицю Users, як користувача.	- username; - password; - pass_copy. -	Виклик методу перевіряє валідність вхідних параметрів. Якщо дані пройшли перевірку, то створюється запит до таблиці бази даних Users для запису нового користувача. Далі йде виклик методу SignUp() для входу нового користувача до системи. Якщо якісь з вхідних параметрів не пройшли перевірку, метод видає користувачеві помилку.

4.9 Інструкція користувача

Користувач може отримати доступ до web-додатку і всього його функціоналу лише перейшовши за посиланням. Після відкриття програми

користувач бачить сторінку логіну чи реєстрації, що зображено на рисунку 4.9.У цьому вікні користувач має можливість увійти у свій акаунт чи зареєструватися, якщо ще не має акаунту. Email має бути валідним, пароль – не менше ніж 8 символів, має містити цифри та великі літери.

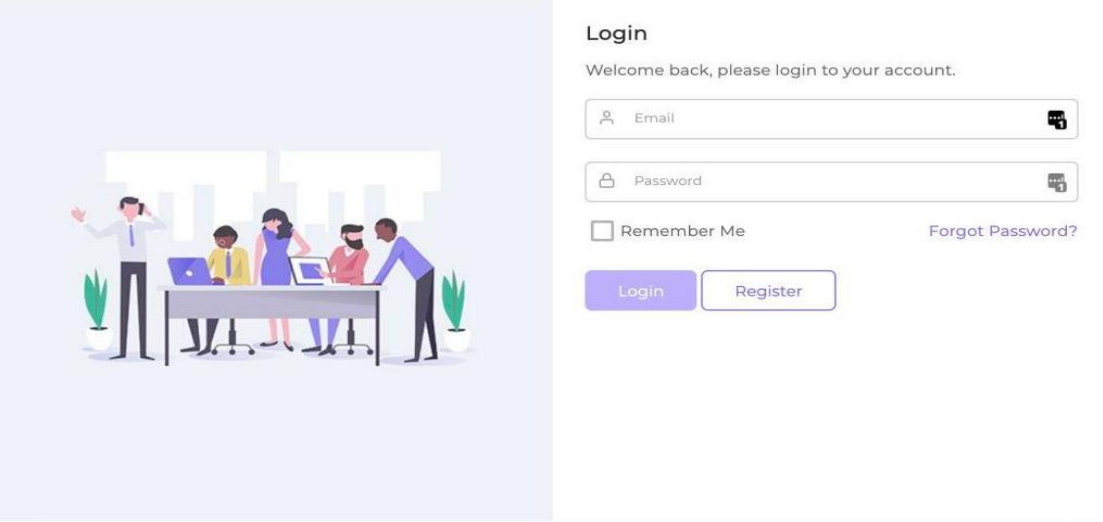

Рисунок 4.9 – Вікно авторизації

Сторінка Profile, що зображена на рисунку 4.9, містить дані про логін, пароль, та локації, які необхідно відображати на карті, як точку відліку, наприклад дім.

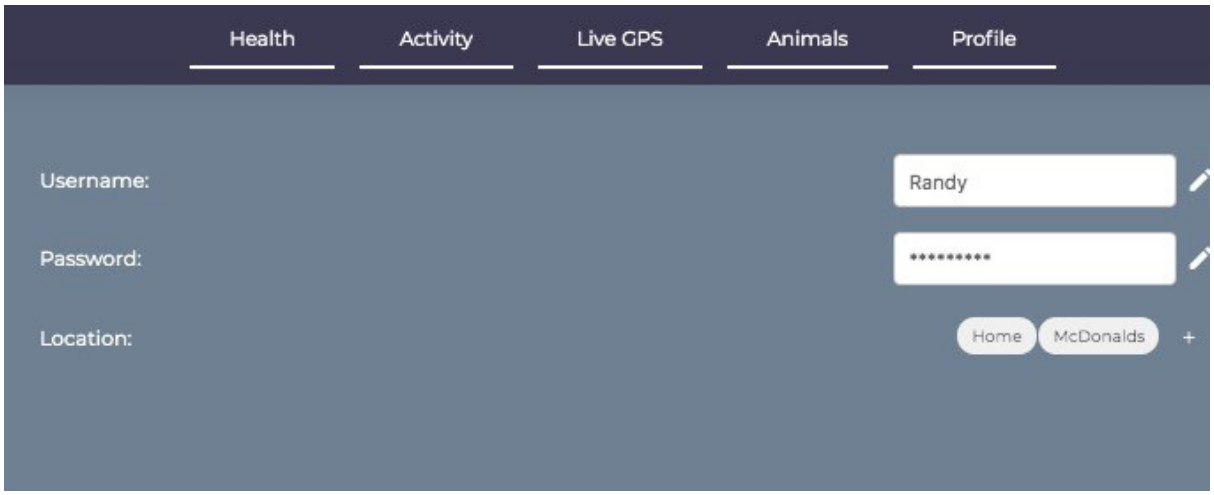

Рисунок 4.10 – Сторінка Profile

Для створення нової локації необхідно натиснути на кнопку “+” та заповнити форму даними про безпечні локації, що в подальшому будуть відображатись на мапі. Форма створення нової локації зображено на рисунку 4.11.

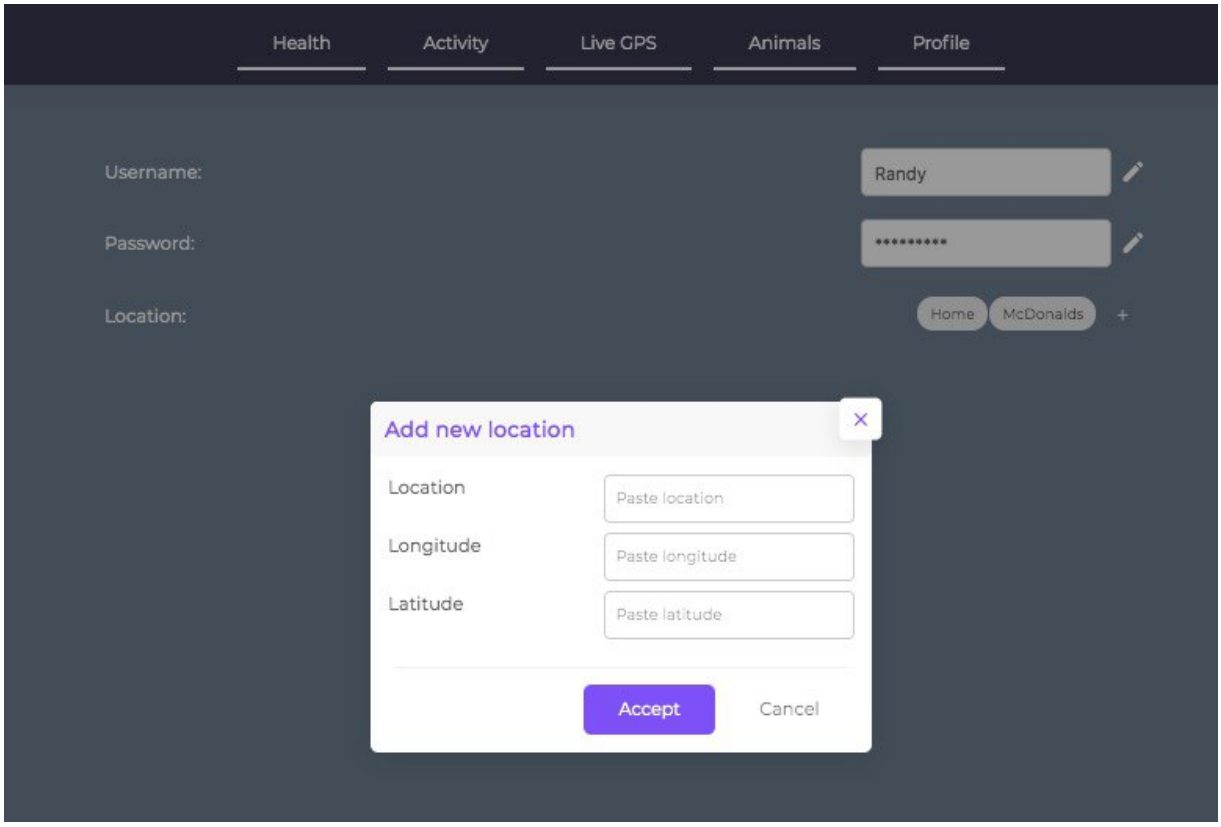

Рисунок 4.11 – Форма створення нової локації

Після реєстрації та входу в додаток користувачу відображається сторінка Animals - рисунок 4.12, в якій можна додавати нових тварин, шляхом натискання кнопки « Add new animal».

Кожен окремий блок даних про тварин дозволяє редагувати та видаляти тварин. А також містить кнопку GPS, яка дозволяє відслідковувати поточне місцезнаходження обраної тварини на сторінці Live GPS.

Основні параметри при додаванні тварини: тип, номер ошийника, ім’я, вага та вік. На рисунку 4.13 зображено вікно додавання тварини на сторінці Animals.

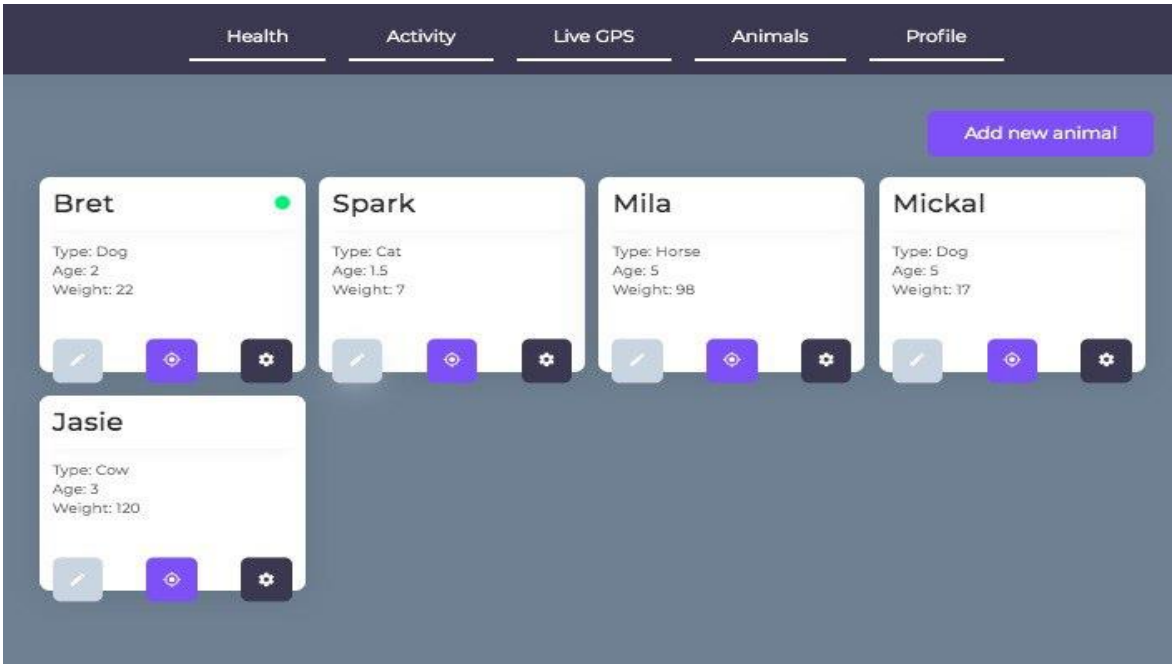

Рисунок 4.12 – Сторінка Animals

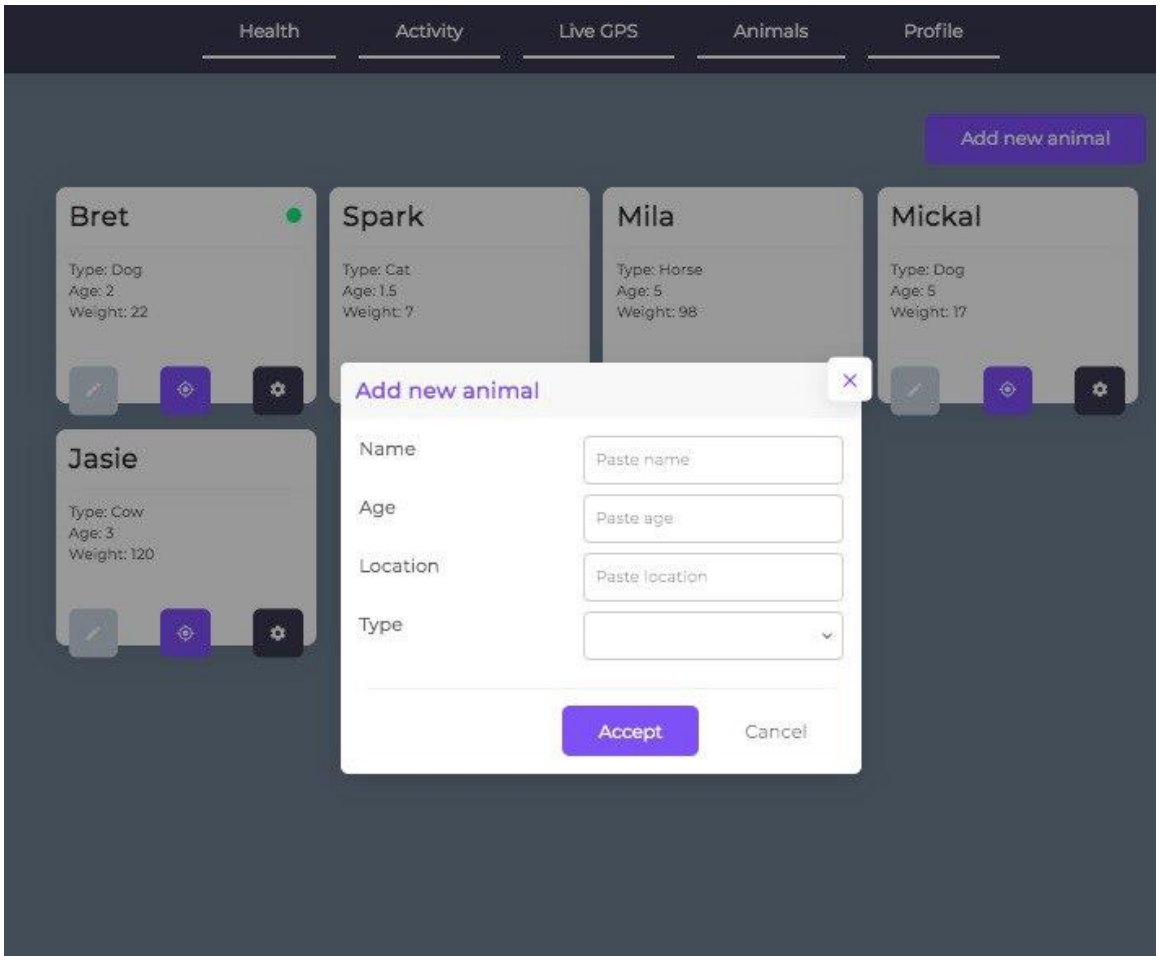

Рисунок 4.13 – Вікно додавання тварини

Щоб переглянути розміщення тварин відносно заданої локації необхідно перейти на сторінку Live GPS, що зображена на рисунку 4.14. Дане вікно відображає поточне розташування всіх тварин та додані користувачем безпечні точки.

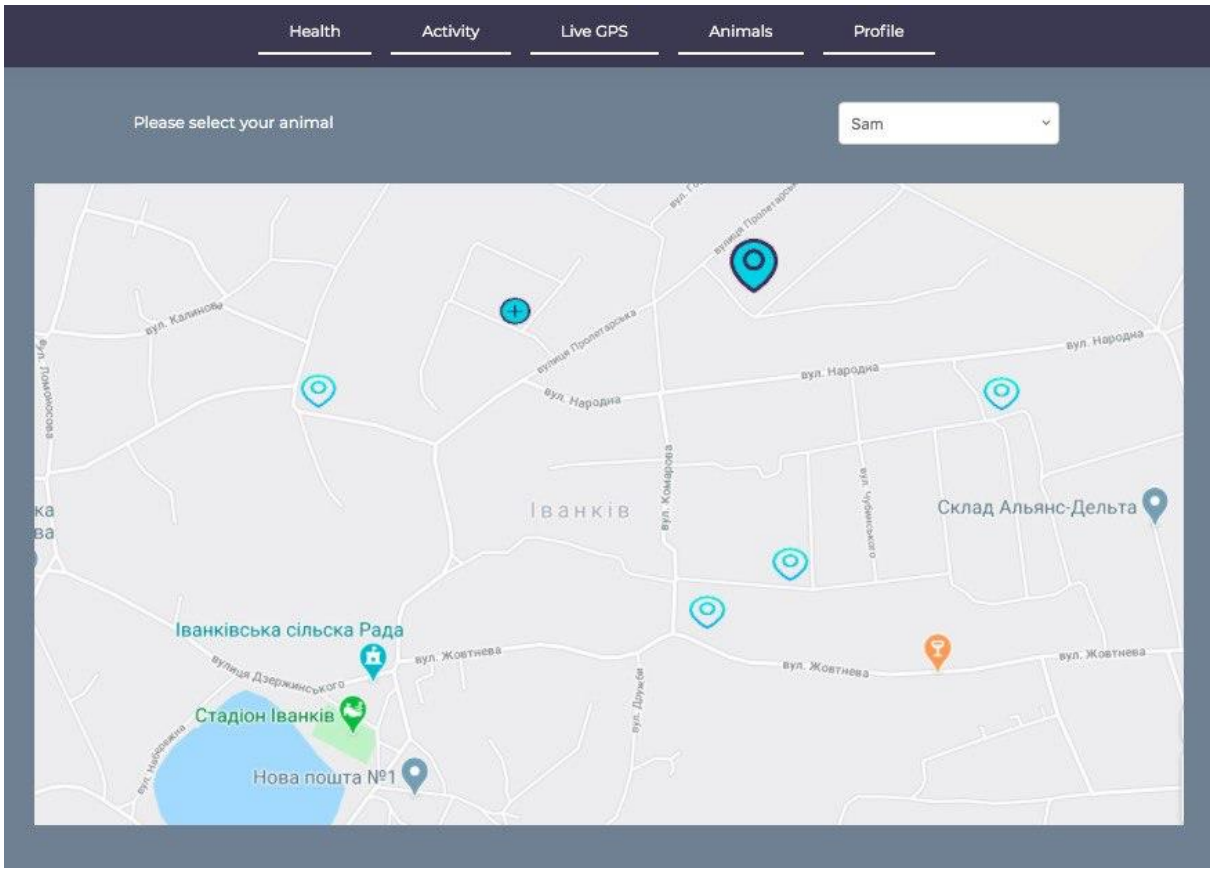

Рисунок 4.14 – Сторінка Live GPS

Життєві показники тварини відображаються на сторінці Health. В кругових діаграмах виведені основні показники обраної тварини - пульс та температура. Нижче згідно визначеному користувачем діапазону дат будується графік зміни життєвих показнизників. Приклад відображення сторінки Health приведено на рисунку 4.15.

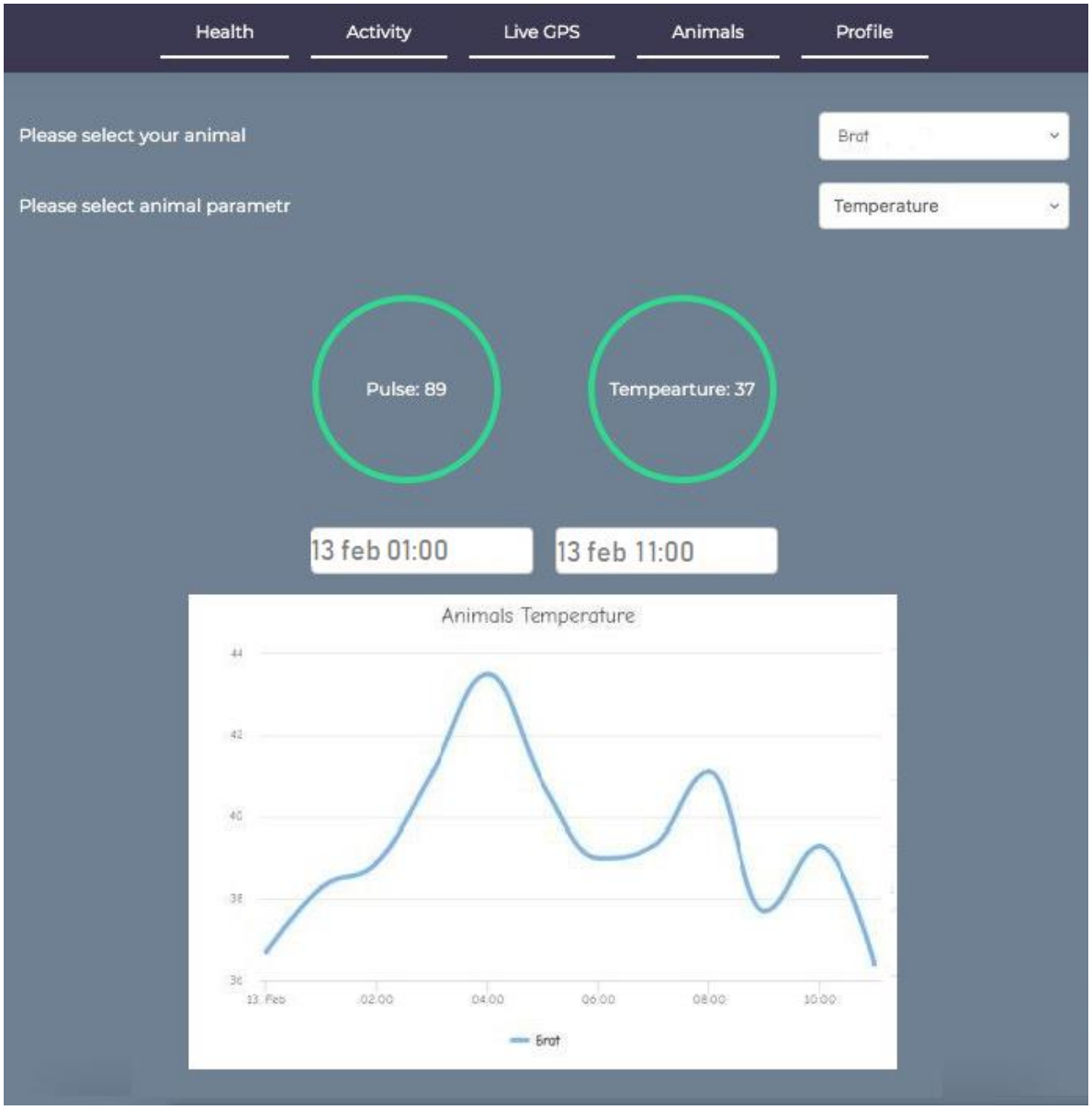

Рисунок 4.15 – Сторінка Health

Також сторінка підтримує множинний вибір тварин, що в свою чергу дає можливість побудувати порівняльних графік відносно обраного діапазону часу та життєвої характеристики. Приклад множинного вибору критеріїв приведено на рисунку 4.16.

Вікно визначення періоду має досить зручний формат – випадаючий календар, як зображено на рисунку 4.17.

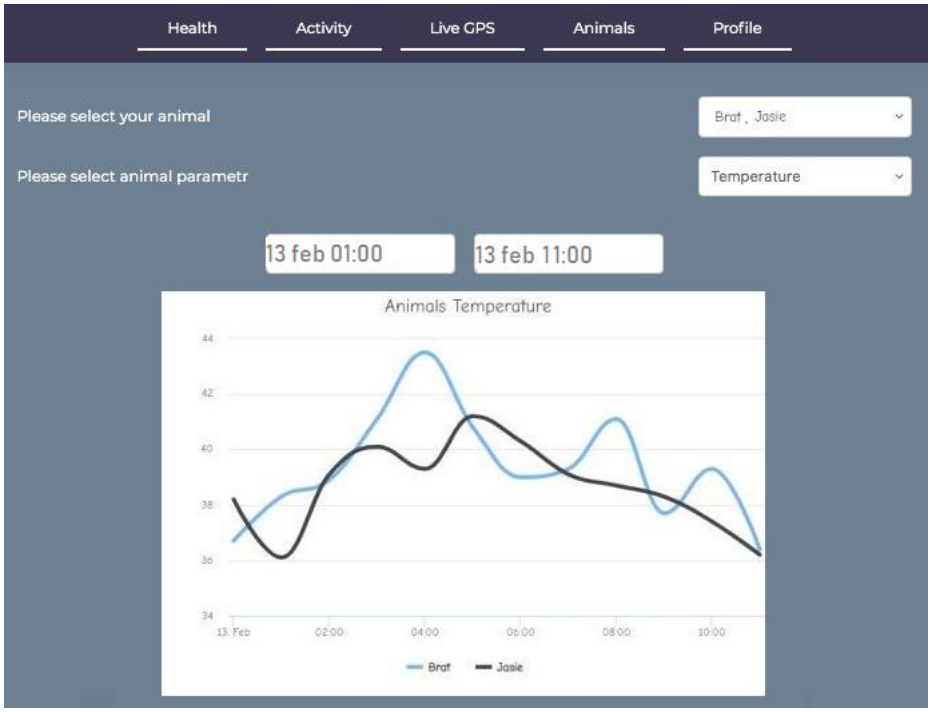

Рисунок 4.16 – Множинний критерій відображення графіку

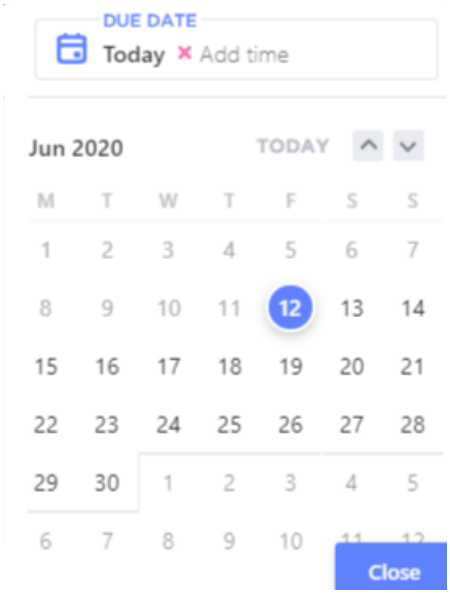

Рисунок 4.17 – Календар для вибору режиму перегляду

Сторінка Activity відображає активність обраної тварин або тварини за обраний період, як зображено на рисунку 4.18.

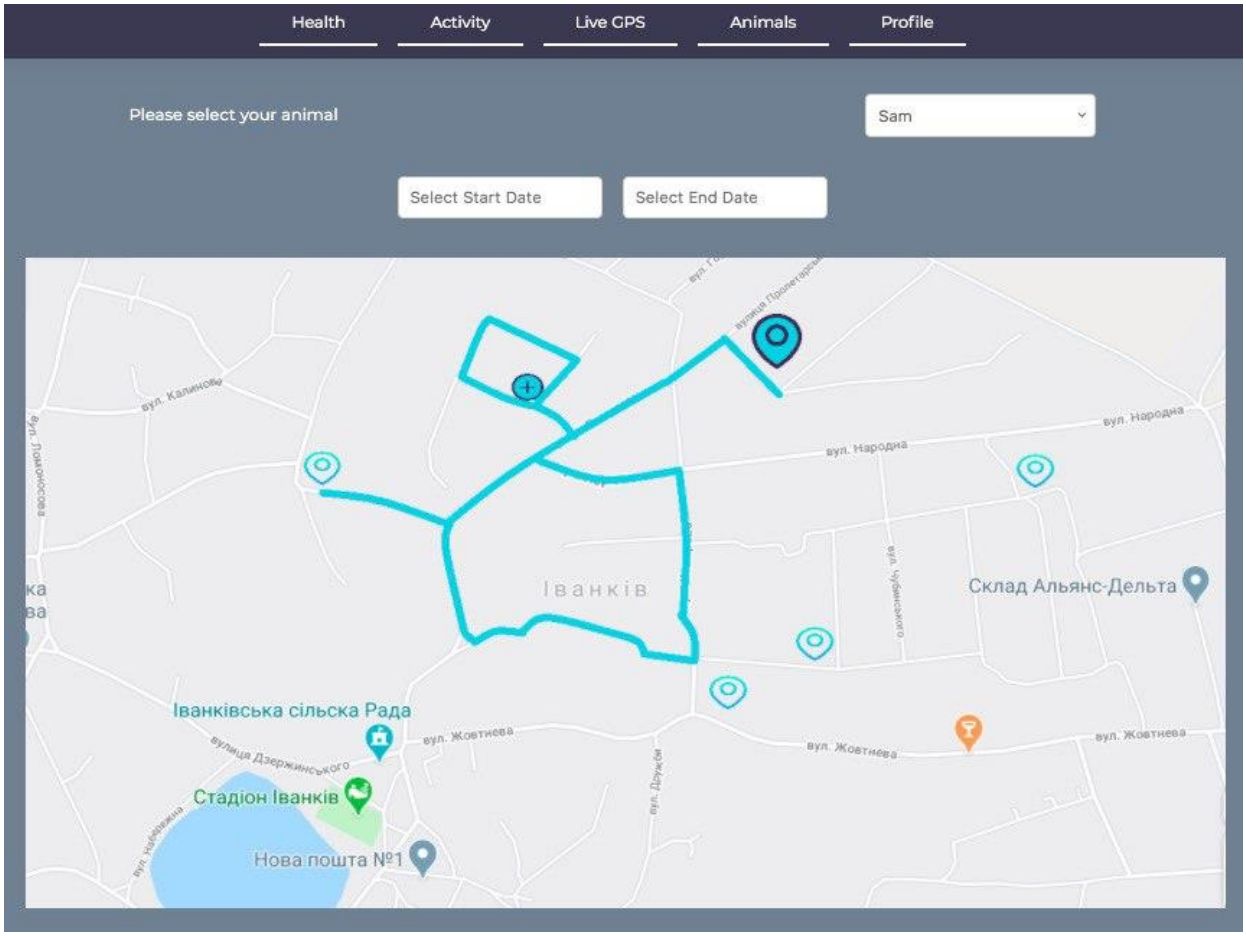

Рисунок 4.18 – Сторінка Activity

Висновки до розділу

У даному розділі було описано ідею та охарактеризовано мову та платформу розробленого додатку. Розглянуто та пояснено роботу серверної частини. Розроблено інструкцію з користування веб-застосунком, в якій почергово описано алгоритм взаємодії користувача з системою. В розділі наведені детальні рисунками системи, що наглядно демонструють роботу додатку.

ВИСНОВКИ

В даному дипломному проєкті було створено покращену системи моніторингу за станом домашніх тварин. Основними параметрами які відстежує система є температура, пульс та місцезнаходження. Згідно поставленого завдання було розроблено структурну схему системи, яка відобразила в собі основні елементи системи та позначені зв'язки між ними. На основі структурної схеми було розроблено принципіальну схему, на якій зображені необхідні елементи та датчики системи. А також було розроблено модель веб-додатку, яка демонструє зібрані системою дані у зручному для користувача форматі

В ході виконання даного дипломного проекту було проаналізовано предметну область розробки, а саме системи моніторингу за станом домашніх тварин PetPace Smart Collar та FitBark, та виділено найкращі характеристики існуючих рішень сьогодення. Згідно створеної порівняльної таблиці, було обрано головні необхідні параметри для розробки покращеної системи.

Для розробки системи було проведено підбір окремих елементів системи: акселерометр та гіроскоп MPU-6050, мікроконтролер Microchip Technology ATMEGA32-16MU, цифровий датчик температури DS18B20, GSM модуль SIM900D. Як елемент живлення було запропоновано використати термоелектричну батарею засновану на ефекті Заєбека, що дозволило зробити систему повністю автономною.

Обрано мову програмування Java Script, яка найкраще підходить для реалізації поставлених задач. Програму було розроблено за допомогою патернів Abstract factory, Proxy, Composite, Model-ViewViewModel.

Для back-end розробки були використані бібліотеки Express.js , Passport.js, Sequelize. Для коректної роботи додатку розгорнуто роботу програмного продукту на сервері з використанням платформи Node.js.

Розроблений сервер реалізує весь необхідний функціонал незалежно від інтерфейсу, тобто впровадження нового функціоналу ніяк не впливатиме на вже існуючий функціонал.

Для front-end розробки були використані бібліотеки Vuetify та Apexcharts.

Також була розроблена структурна база даних MySQL, яка відповідає усім необхідним потребам розробленого додатку. Створено та описано детальну інструкцію користувача, на всіх етапах використання веб застосунку

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Express - фреймворк веб-приложений Node.js [Електронний ресурс] – Доступ: https://expressjs.com/ru/
2. Simple, unobtrusive authentication for Node.js [Електронний ресурс] – Доступ: http://www.passportjs.org/
3. Sequelize ORM [Електронний ресурс] – Доступ: https://sequelize.org/
4. The Progressive JavaScript Framework Vue.js [Електронний ресурс] – Доступ: https://vuejs.org/
5. R. N. Handcock et al., “Monitoring animal behaviour and environmental interactions using wireless sensor networks, GPS collars and satellite remote sensing,” Sensors, vol. 9, no. 5, pp. 3586–3603, 2009.
6. S.-J. Jung and W.-Y. Chung, “Non-intrusive healthcare system in global machine-to-machine networks,” IEEE Sensors J., vol. 13, no. 12, pp. 4824–4830, Dec. 2013.
7. K. Smith, A. Martinez, R. Craddolph, H. Erickson, D. Andersen, and S. Warren, “An integrated cattle health monitoring system,” in Proc. IEEE Annu. Int. Conf. EMBS, New York, NY, USA, Aug./Sep. 2006, pp. 4659–4662.
8. Wikipedia [Електронний ресурс] – Доступ: https://uk.wikipedia.org/wiki/BPMN
9. W. D. Jones, “Taking body temperature, inside out [body temperature monitoring],” IEEE Spectr., vol. 43, no. 1, pp. 13–15, Jan. 2006.
10. Гэри Маклин Холл Адаптивный код: гибкое кодирование с помощью паттернов проектирования – Диалектика-Вильямс, 2019 - 23с

11. T. K. Hamrita, E. W. Tollner, and R. L. Schafer, “Toward fulfilling the robotic farming vision: Advances in sensors and controllers for agricultural applications,” IEEE Trans. Ind. Appl., vol. 36, no. 4, pp. 1026–1032, Jul./Aug. 2000.

12. Y. Kim, R. G. Evans, and W. M. Iversen, “Remote sensing and control of an irrigation system using a distributed wireless sensor network,” IEEE Trans. Instrum. Meas., vol. 57, no. 7, pp. 1379–1387, Jul. 2008.

13. B. Wietrzyk and M. Radenkovic, “Enabling large scale ad hoc animal welfare monitoring,” in Proc. 5th Int. Conf. Wireless Mobile Commun. (ICWMC), Cannes, France, Aug. 2009, pp. 401–409.

14. Y. Guo, P. Corke, G. Poulton, T. Wark, G. Bishop-Hurley, and D. Swain, “Animal behaviour understanding using wireless sensor networks,” in Proc. IEEE Int. Conf., Nov. 2006, pp. 607–614, ISBN: 1-4244-0419-3.

15. D. Wobschall, “Networked sensor monitoring using the universal IEEE 1451 standard,” IEEE Instrum. Meas. Mag., vol. 11, no. 2, pp. 18–22, Apr. 2008.

16. A. Kumar and G. P. Hancke, “Energy efficient environment monitoring system based on the IEEE 802.15.4 standard for low cost requirements,” IEEE Sensors J., vol. 14, no. 8, pp. 2557–2566, Aug. 2014.

ДОДАТОК А

Клас LiveGPS

```
1.  const { Location } = require('../models')
2.  const { jwtGenerator } = require('../helpers')
3.  module.exports = {
4.    // each call 10000ms done
5.    async getLiveLocation (req, res) {
6.      try {
7.        const liveGPS = await Location.get(req.body)
8.        const location = await Location.get(res.body.animal_id)
9.        res.send({
10.         animal: animal_id,
11.         live: liveGPS,
12.         location: location,
13.         token: jwtGenerator.generate(userJson)
14.       })
15.     } catch (err) {
16.       res.status(400).send({
17.         error: ${err}
18.       })
19.     }
20.   }
```

Клас Location

```
1.  const { Activity } = require('../models')
2.  const { Location } = require('../models')
3.  const { jwtGenerator } = require('../helpers')
4.  module.exports = {
5.    async getLocation (req, res) {
6.      try {
```

```
7.        const activity = await Activity.get(req.body)
8.        const location = await Location.get(res.body.animal_id)
9.        res.send({
10.         animal: animal_id,
11.         activity: activity,
12.         location: location,
13.         tips: location.checkPoints,
14.         token: jwtGenerator.generate(userJson)
15.       })
16.     } catch (err) {
17.       res.status(400).send({
18.         error: ${err}
19.       })
20.     }
21.   }
```

Клас Health

```
1.  const { Health } = require('../models')
2.  const { Pulse } = require('../models')
3.  const { Temperature } = require('../models')
4.  const { jwtGenerator } = require('../helpers')
5.  module.exports = {
6.    async getInfo (req, res) {
7.      try {
8.        const health = await Health.get(req.body)
9.        const pulse = await Pulse.get(req.body)
10.       const temperature = await Temperature(req.body)
11.       const healthJson = health.toJSON()
12.       res.send({
13.         animal: animal_id,
```

```
14.         pulse: pulse,
15.         temperature: temperature,
16.         token: jwtGenerator.generate(userJson)
17.       })
18.     } catch (err) {
19.       res.status(400).send({
20.         error: ${err}
21.       })
22.     }
23.   }
```

Клас Animal

```
1.   const { Animal } = require('../models')
2.   module.exports = {
3.     async create_animal (req, res) {
4.       try {
5.         const animal = await Animal.create(req.body)
6.         const animalJson = animal.toJSON()
7.         res.send({
8.           data: animalJson
9.         })
10.      } catch (err) {
11.        res.status(400).send({
12.          error: ${err}
13.        })
14.      }
15.    },
16.    async remove_animal (req, res) {
17.      console.log(req.body)
18.      try {
```

```
19.         await Animal.destroy({
20.           where: {
21.             id: req.body.id
22.           }
23.         })
24.         res.send({
25.           data: ${req.body}, succes
26.         })
27.       } catch (err) {
28.         res.status(400).send({
29.           error: ${err}
30.         })
31.       }
32.     },
33.     async getAll (req, res) {
34.       try {
35.         console.log(req.body)
36.         const animal = await Animal.findAll()
37.         // const animalJson = animal.toJSON()
38.         res.send({
39.           data: animal
40.         })
41.       } catch (err) {
42.         res.status(400).send({
43.           error: ${err}
44.         })
45.       }
46.     }
47.   }
```

Клас User

```
1.  const { User } = require('../models')
2.  const { jwtGenerator } = require('../helpers')
3.  module.exports = {
4.    async register (req, res) {
5.      try {
6.        const user = await User.create(req.body)
7.        const userJson = user.toJSON()
8.        res.send({
9.          user: userJson,
10.         token: jwtGenerator.generate(userJson)
11.       })
12.     } catch (err) {
13.       res.status(400).send({
14.         error: ${err}
15.       })
16.     }
17.   },
18.
19.   async login (req, res) {
20.     try {
21.       const { email, password } = req.body
22.       const user = await User.findOne({
23.         where: {
24.           email: email
25.         }
26.       })
27.       if (!user) {
28.         return res.status(403).send({
29.           error: 'Check your login please'
```

```
30.           })
31.         }
32.         const isPasswordValid = await
33.   user.comparePassword(password)
34.         if (isPasswordValid) {
35.           return res.status(403).send({
36.             error: 'Password is invalid for this email'
37.           })
38.         }
39.         const userJson = user.toJSON()
40.         res.send({
41.           user: userJson,
42.           token: jwtGenerator.generate(userJson)
43.         })
44.       } catch (err) {
45.         console.log('ERROR IS HERE', err)
46.         res.status(400).send({
47.           error: 'Invalid login info'
48.         })
49.       }
50.     }
51.   }
```